Microbial residence time is a controlling parameter of the taxonomic composition and functional profile of microbial communities

Mansfeldt, Cresten ; Achermann, Stefan ; Men, Yujie ; Walser, Jean-Claude ; Villez, Kris ; Joss, Adriano ; Johnson, David R ; Fenner, Kathrin

Abstract: A remaining challenge within microbial ecology is to understand the determinants of richness and diversity observed in environmental microbial communities. In a range of systems, including activated sludge bioreactors, the microbial residence time (MRT) has been previously shown to shape the microbial community composition. However, the physiological and ecological mechanisms driving this influence have remained unclear. Here, this relationship is explored by analyzing an activated sludge system fed with municipal wastewater. Using a model designed in this study based on Monod-growth kinetics, longer MRTs were shown to increase the range of growth parameters that enable persistence, resulting in increased richness and diversity in the modeled community. In laboratory experiments, six sequencing batch reactors treating domestic wastewater were operated in parallel at MRTs between 1 and 15 days. The communities were characterized using both 16S ribosomal RNA and non-target messenger RNA sequencing (metatranscriptomic analysis), and model-predicted monotonic increases in richness were confirmed in both profiles. Accordingly, taxonomic Shannon diversity also increased with MRT. In contrast, the diversity in enzyme class annotations resulting from the metatranscriptomic analysis displayed a non-monotonic trend over the MRT gradient. Disproportionately high abundances of transcripts encoding for rarer enzymes occur at longer MRTs and lead to the disconnect between taxonomic and functional diversity profiles.

DOI: https://doi.org/10.1038/s41396-019-0371-6

Supplementary Figures

Table of Contents

Supplemental Figure 13. Bitscore histograms representing the quality of the DIAMOND annotation for TP1R1 with default DIAMOND values against the (a) Uniprot EC library only (14,871,396 sequences, 5,767,210,689 amino acids, with an average of 387.8 a.a., downloaded on March 6th, 2018) and (b) the full Uniprot library (109,414,541 sequences, 36,814,708,888 amino acids, average length of protein 336.5 a.a., downloaded on March 6th, 2018).

Supplemental Figure 14. The range of μ_{max} and K_s parameters for the 1 and 15 d MRT Sequencing Batch Reactor models when allowing only the μ_{max} and K_s parameters to vary (the b_e is held constant). For comparison, see Figure 1.b within the main text.

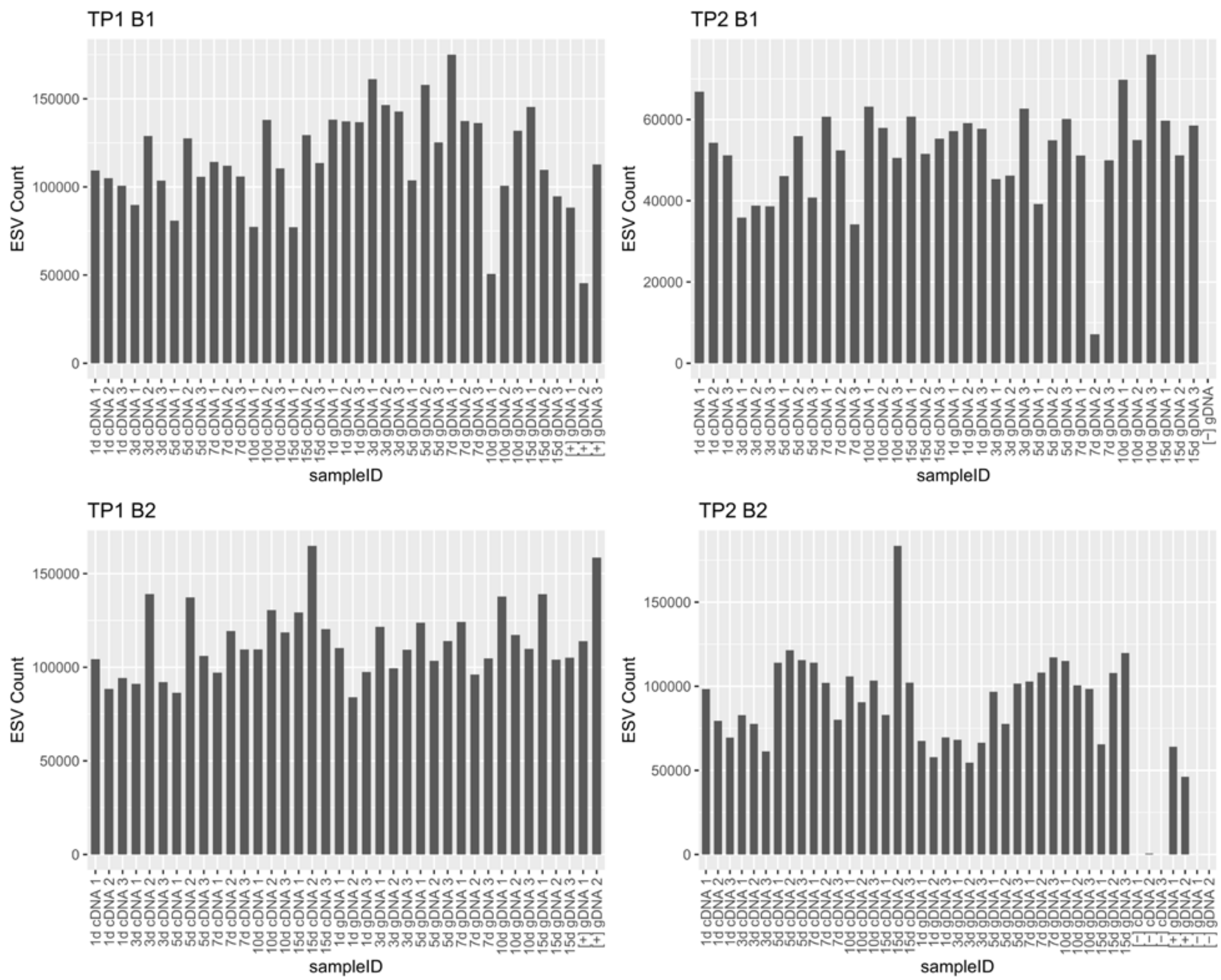
Supplemental Figure 15. A trace of the growth parameter solution path, the Shannon diversity, and the richness results of the model when the of μ_{max} and b_e are constrained (a) by the fastest grower or (b) by the slowest grower.

Supplemental Figure 16. (a,b) The calculated Hill numbers (0-3) for the rarified B1-primer amplified 16S rRNA (black) and rDNA (red) data. (c,d) The abundance data distributed into taxonomic orders; the top 10 of the sums across each time-point are colored, resulting in 15 orders being represented.

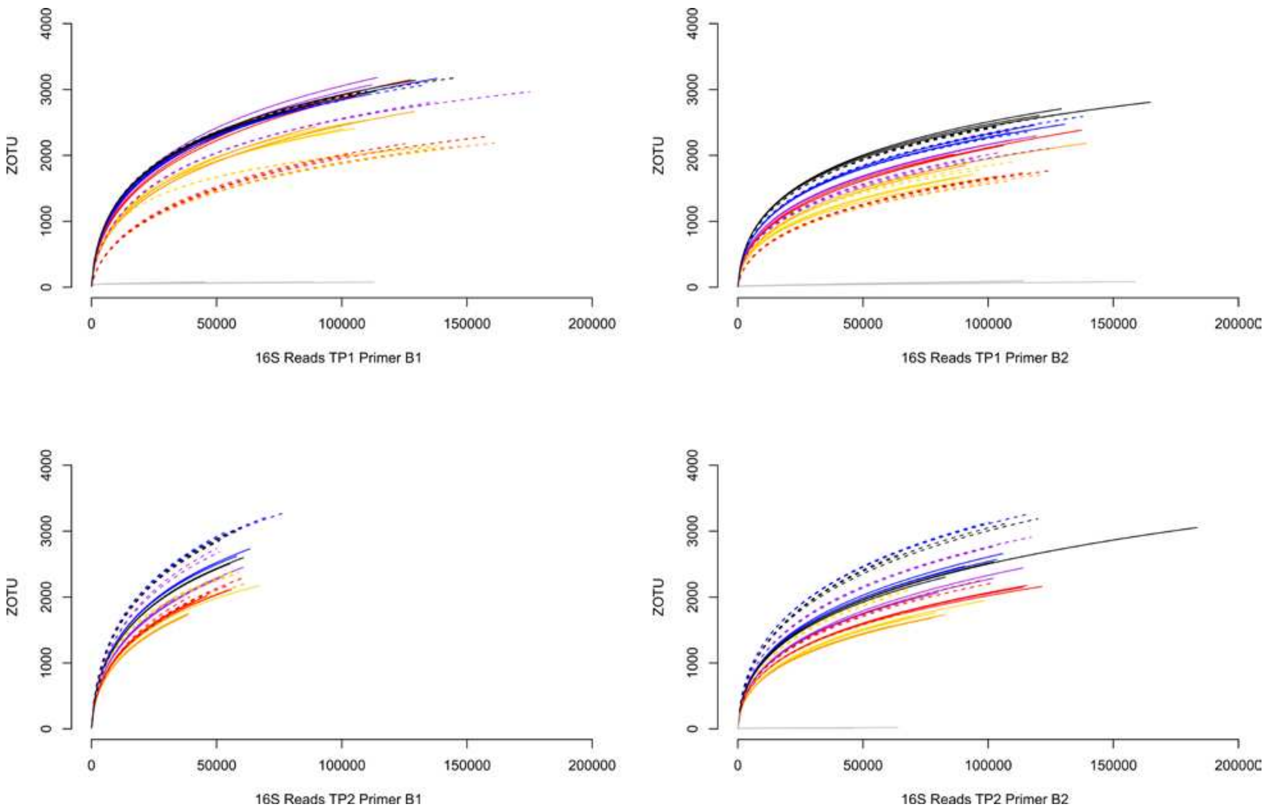
Supplementary Figure 17. Summary of all EC sub-subclasses across the MRT gradient. The heatmaps are organized hierarchically according to a Euclidean distance and ward clustering of the (a) EC sub-subclass fraction of total abundance. (b) The 15/1d MRT abundance log ratio. The within EC sub-subclass taxonomic (c) Hill 0 richness and (d) Hill 1 diversity metrics were calculated based on the Uniport identifiers of the organism of origin that provided the annotation to the reads. (e) The fraction of the total reads that were annotated per EC sub-subclass originating from a Eukaryotic sequence in the Uniprot database.

Supplementary Figure 18. Summary of the top 50 maximum fractional abundances EC numbers across the MRT gradient. The heatmaps are organized hierarchically according to a Euclidean distance and ward clustering of the (a) EC number fraction of total abundance. (b) The 15/1d MRT abundance log ratio. The within EC number taxonomic (c) Hill 0 richness and (d) Hill 1 diversity metrics were calculated based on the Uniport identifiers of the organism of origin that provided the annotation to the reads. (e) The fraction of the total reads that were annotated per EC number originating from a Eukaryotic sequence in the Uniprot database.

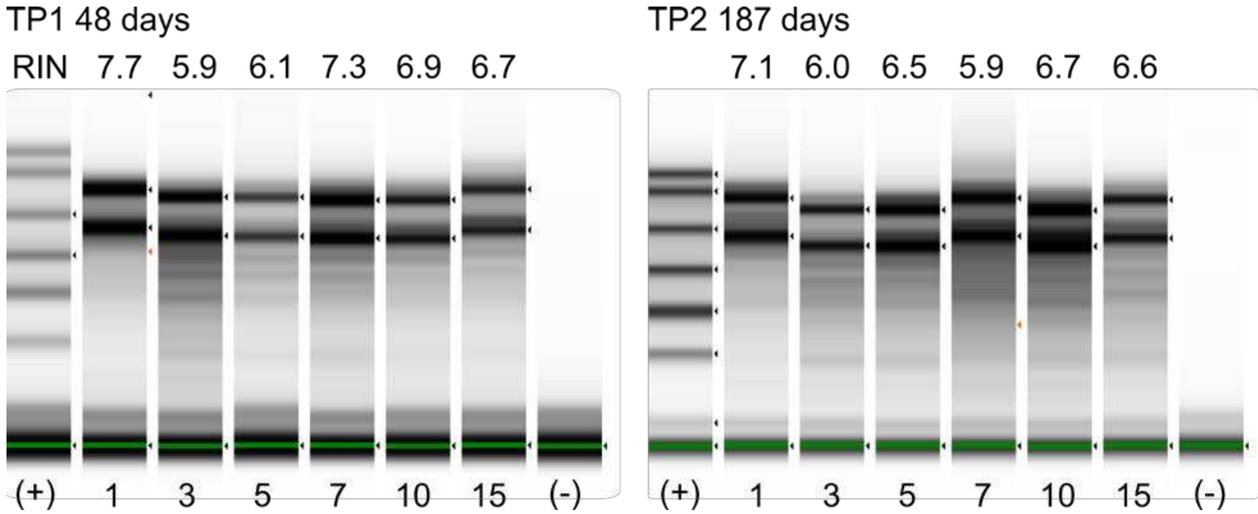
Supplementary Figure 19. The Hill 0 (a) and 1 (b) values for the functionally annotated RNA originating from Bacteria only data binned into common EC number for TP1 (solid) and TP2 (dashed) at the fractional abundance cutoff of 10^{-7}.The Hill 0 and 1 values were calculated for each reactor using the rtk v0.2.5.4 package in R v3.5.1 for 10 bootstrap sub-selections of the annotations. The lines trace the bootstrap mean.



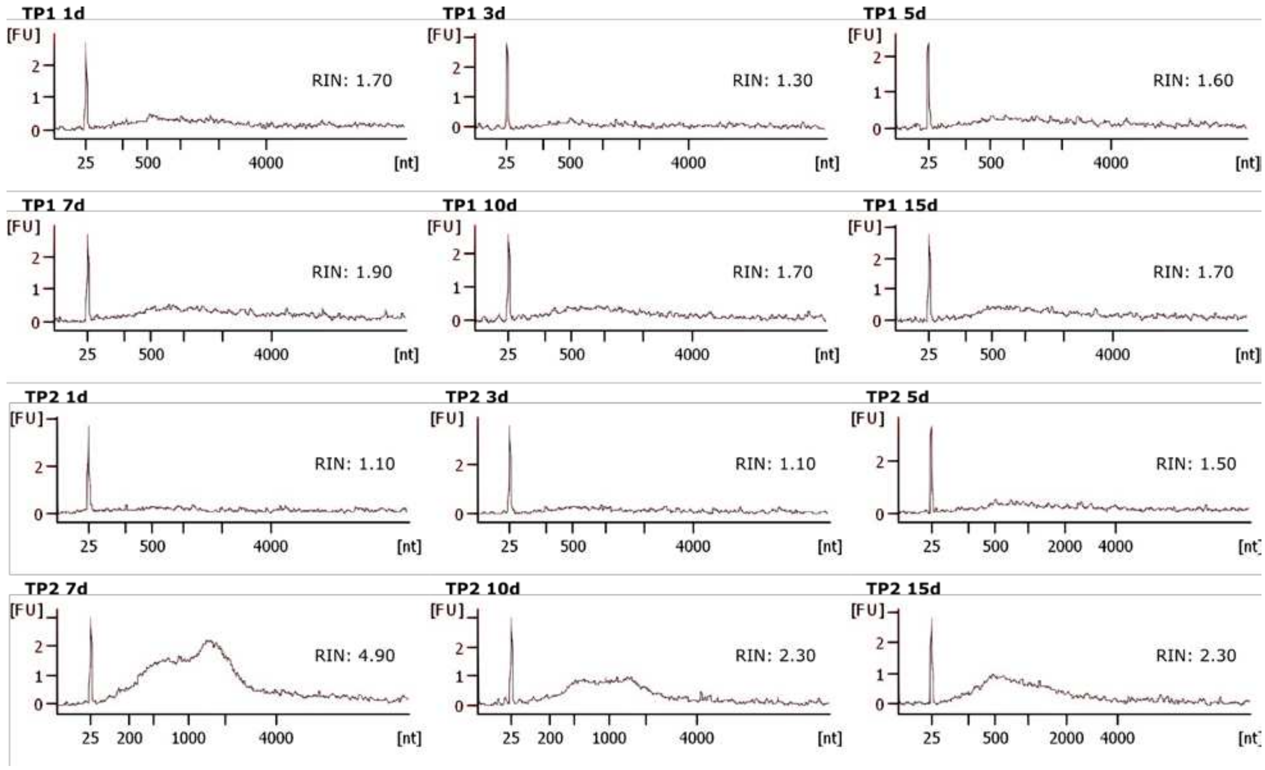
Supplemental Figure 1. The distribution of 16S reads per sequenced libraries (in triplicate) for the two TPs and two primers (B1 and B2). The [+] samples were positive mock communities used to assess both the accuracy of the negative signal and the recovery of an expected distribution. The [-] samples were blanks carried through the PCR and sequencing steps. Note: only TP2 B17d gDNA 2 displayed poor performance, and TP1 B2 [+] gDNA 3 is not displayed because that sample library contained ten-times the reads when compared to the other TP1 B2 samples.



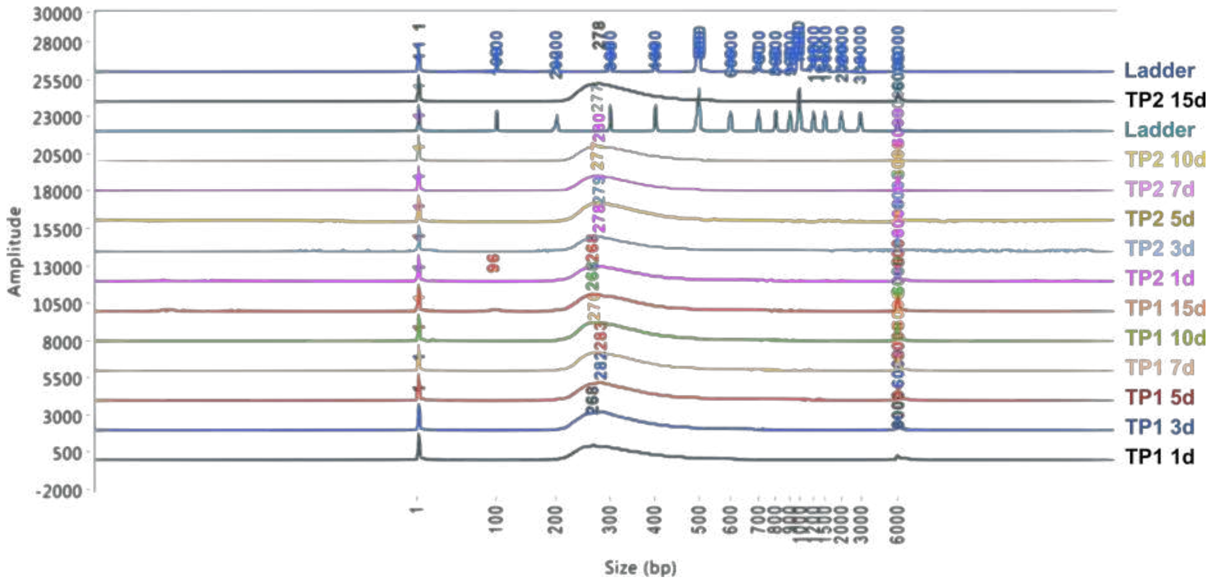
Supplemental Figure 2. Rarefaction curves for the 16S rRNA data. The number of resulting ESVs from triplicate libraries for the 16S rRNA (solid) and rDNA (dashed) 1,3,5,7,10, and 15 d (yellow, orange, red, purple, blue, and black, respectively) reactors are plotted against the sub-selection read count for both Primer B1 and B2. The positive mock communities are plotted in grey.



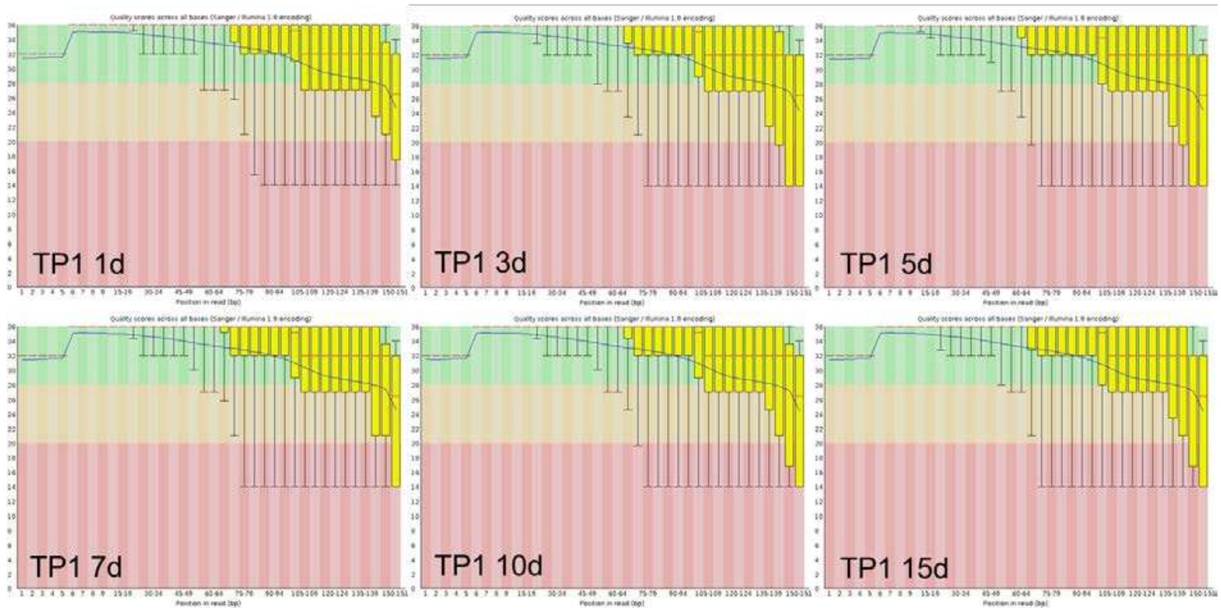
Supplemental Figure 3. Tape Station traces of the RNA pool before rRNA depletion. The numbers below the figure indicate the MRT of the reactor, ladder (+), and empty control (-). The numbers above the figure indicate the RNA Integrity Number (RIN).



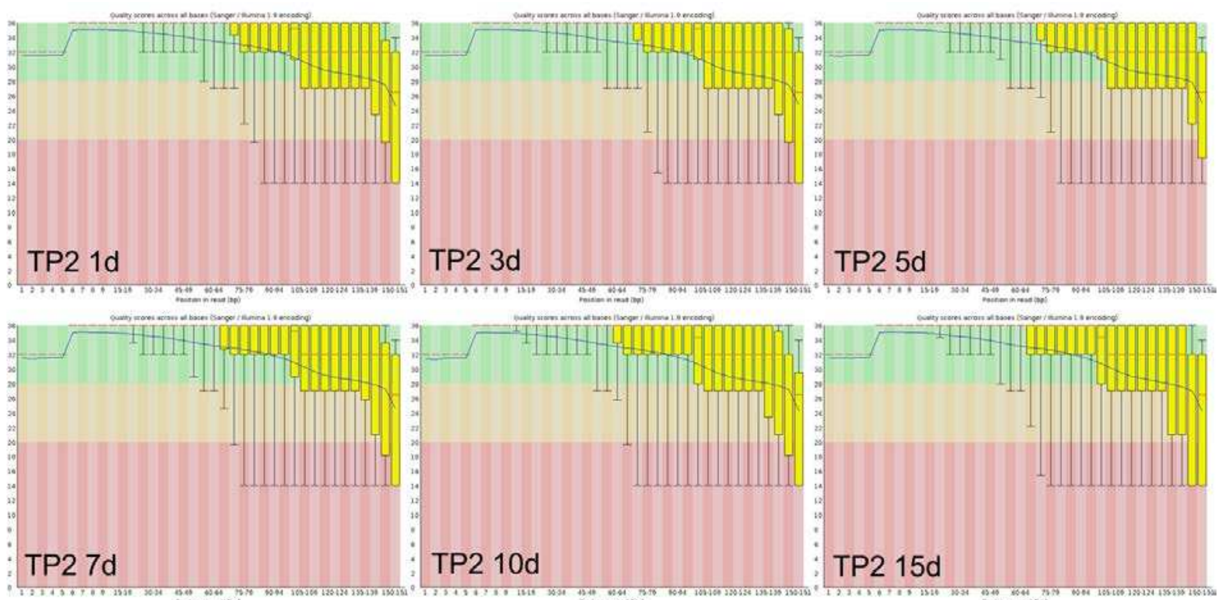
Supplemental Figure 4. Bioanalyzer traces of the RNA pool after rRNA depletion using the RiboZero Epidemiology Kit. RIN indicates the RNA Integrity Number.



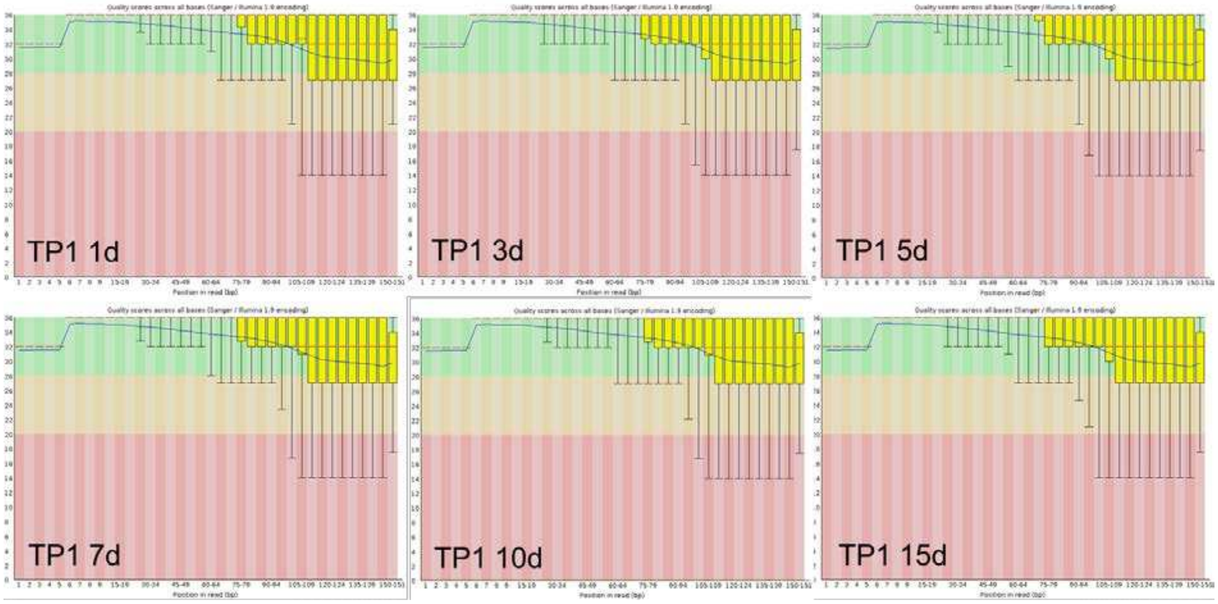
Supplemental Figure 5. Fragment quality of the library run on the Illumina NextSEQ.



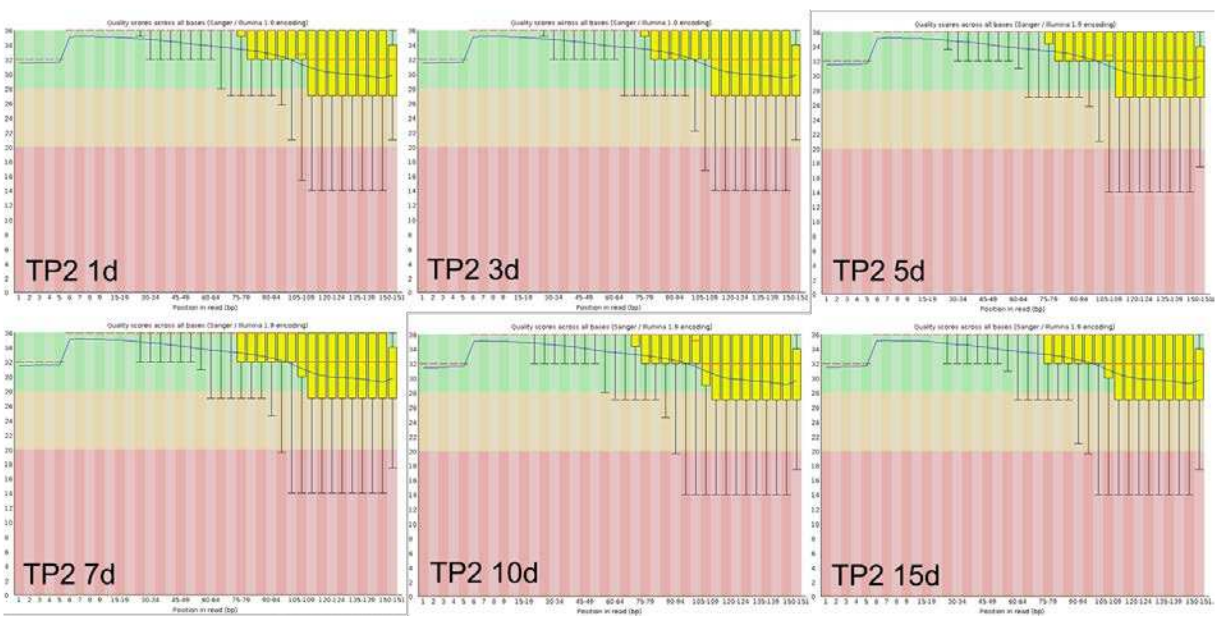
Supplemental Figure 6. TP1 48 day raw FastQC quality results.



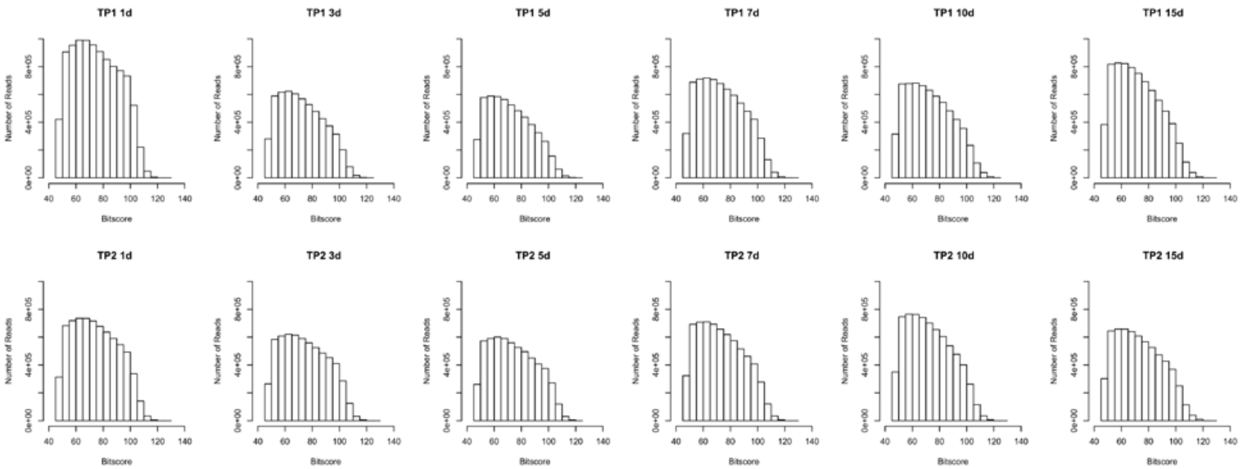
Supplemental Figure 7. TP2 187 day raw FastQC quality results.



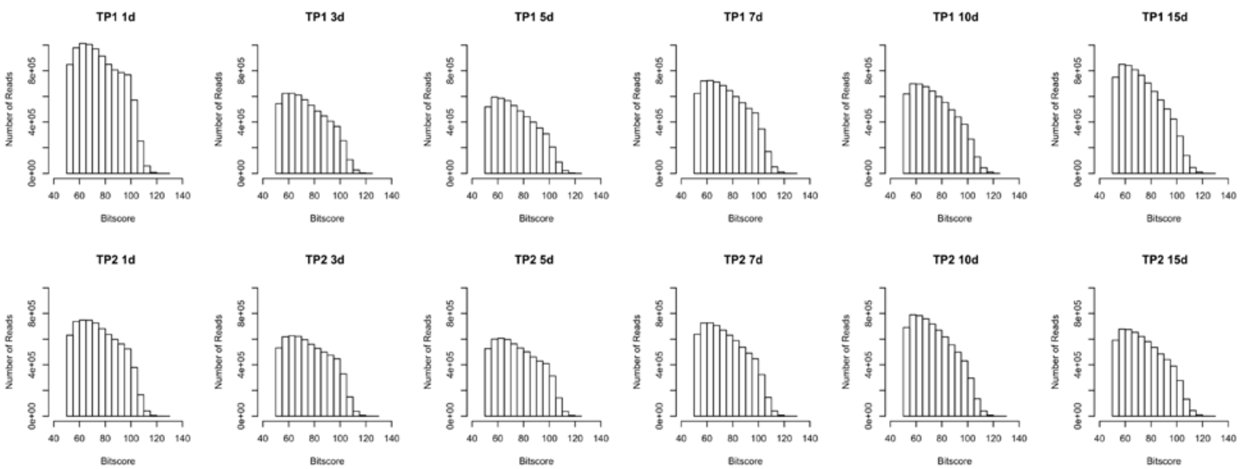
Supplemental Figure 8. TP1 48 day trimmed FastQC quality results.



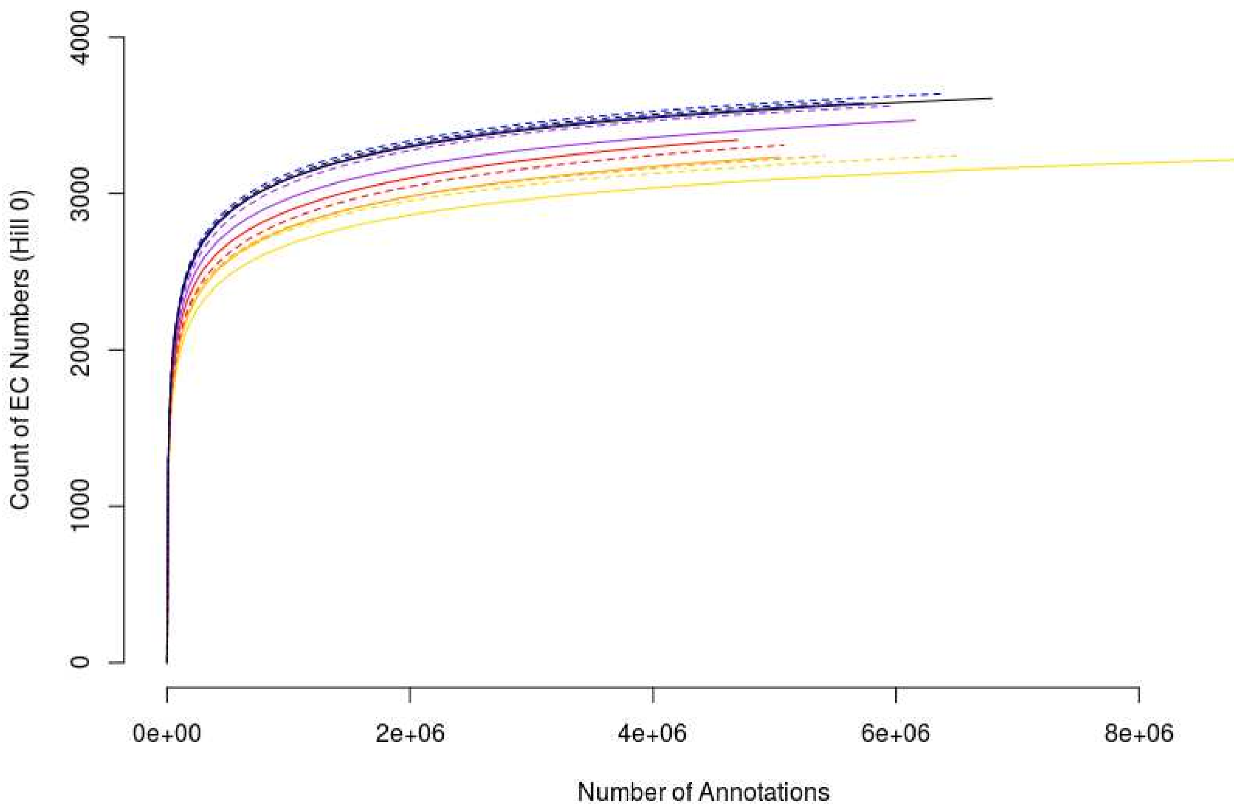
Supplemental Figure 9. TP2 187 day trimmed FastQC quality results.



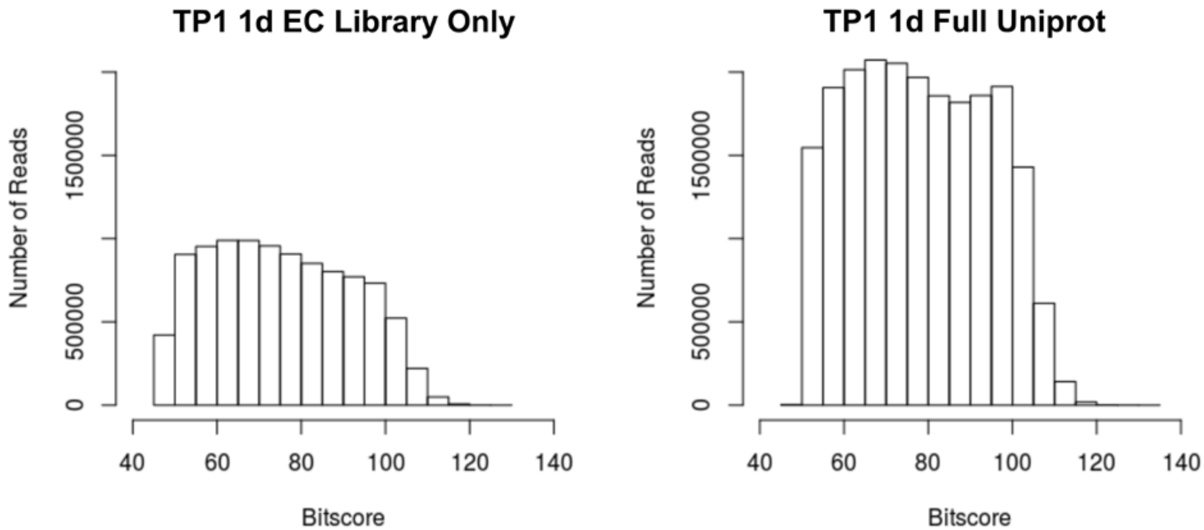
Supplemental Figure 10. Bitscore histograms representing the quality of the DIAMOND annotation for all reactors against the Uniprot EC library only (14,871,396 sequences, downloaded on March 6th, 2018).



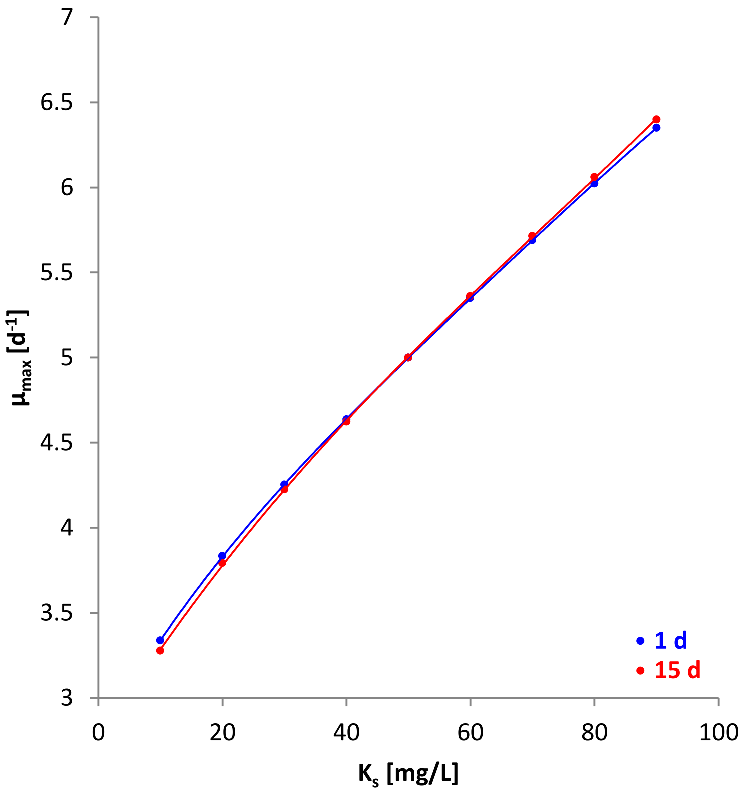
Supplemental Figure 11. Bitscore histograms representing the quality of the DIAMOND with a bitscore cutoff of 50 annotation for all reactors against the Uniprot EC library (14,871,396 sequences, downloaded on March 6th, 2018).



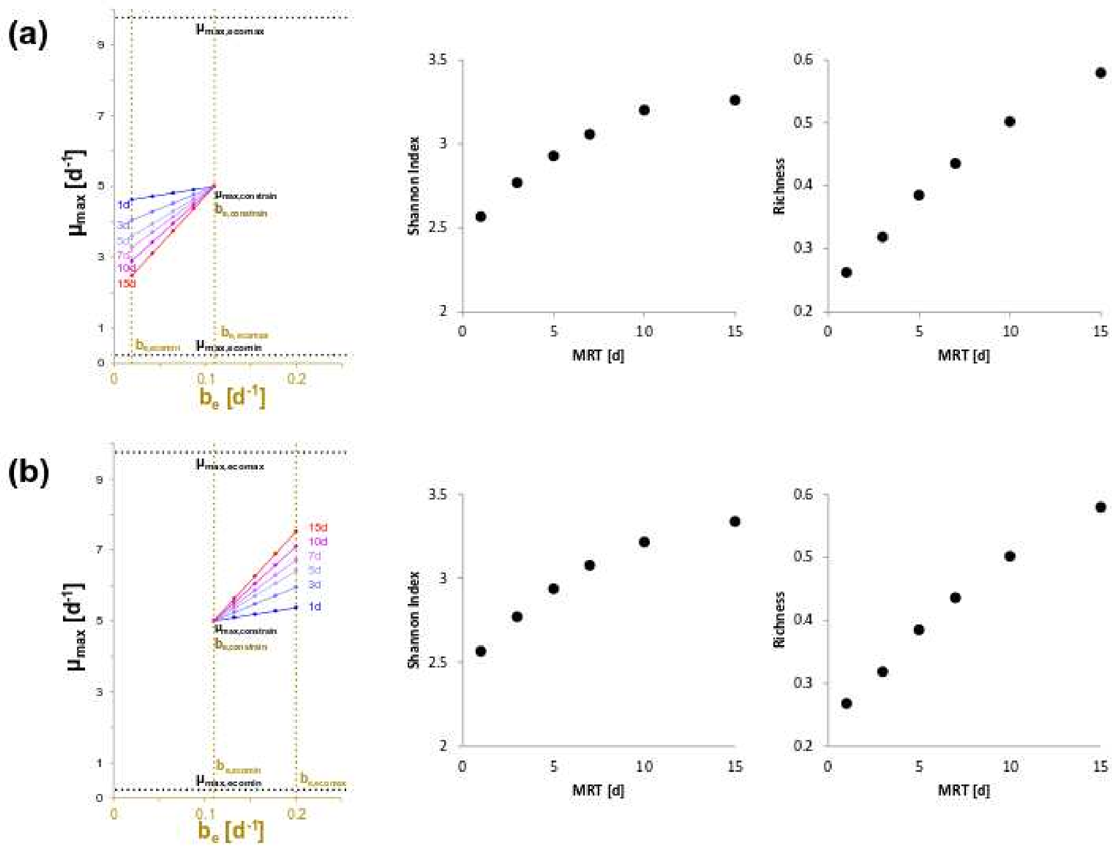
Supplemental Figure 12. Rarefaction curves for the EC number annotations for the TP1 (solid) and TP2 (dashed) 1,3,5,7,10, and 15 d (yellow, orange, red, purple, blue, and black, respectively) reactors.



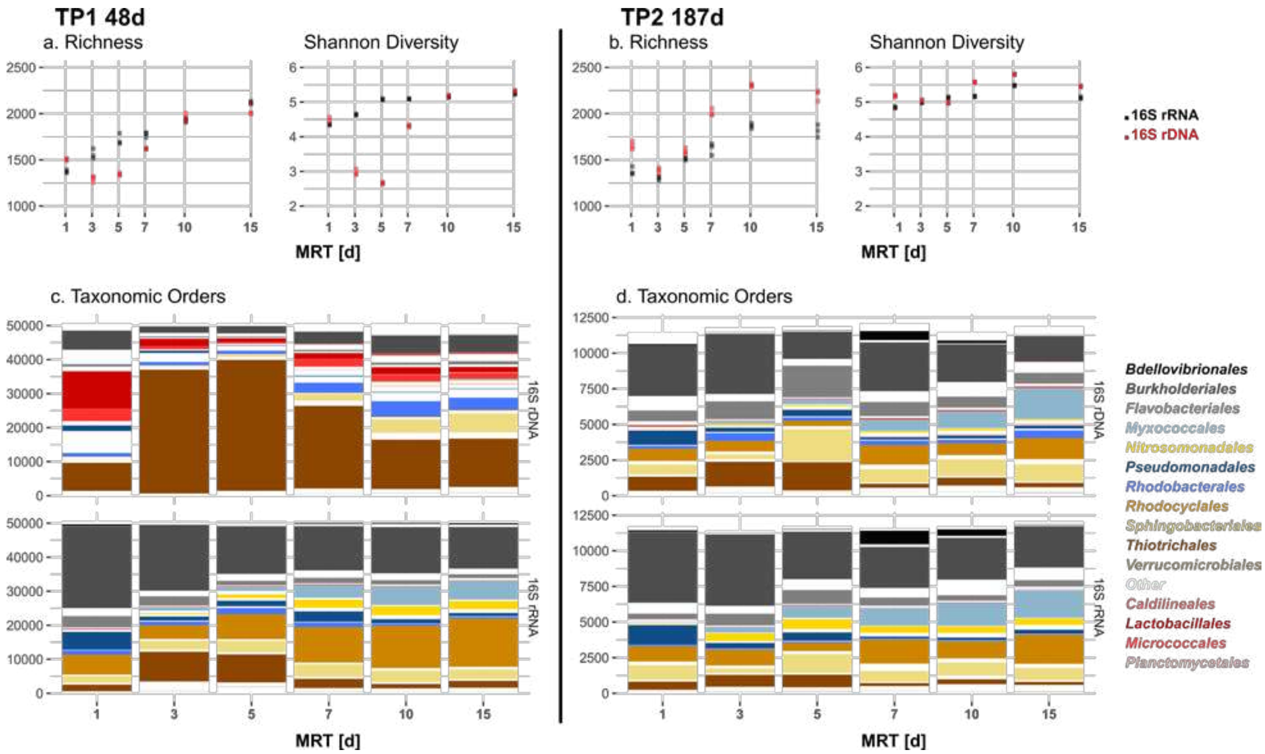
Supplemental Figure 13. Bitscore histograms representing the quality of the DIAMOND annotation for TP1 1 d MRT with default DIAMOND values against the (a) Uniprot EC library only (14,871,396 sequences, 5,767,210,689 amino acids, with an average of 387.8 a.a., downloaded on March 6th, 2018) and (b) the full Uniprot library (109,414,541 sequences, 36,814,708,888 amino acids, average length of protein 336.5 a.a., downloaded on March 6th, 2018).



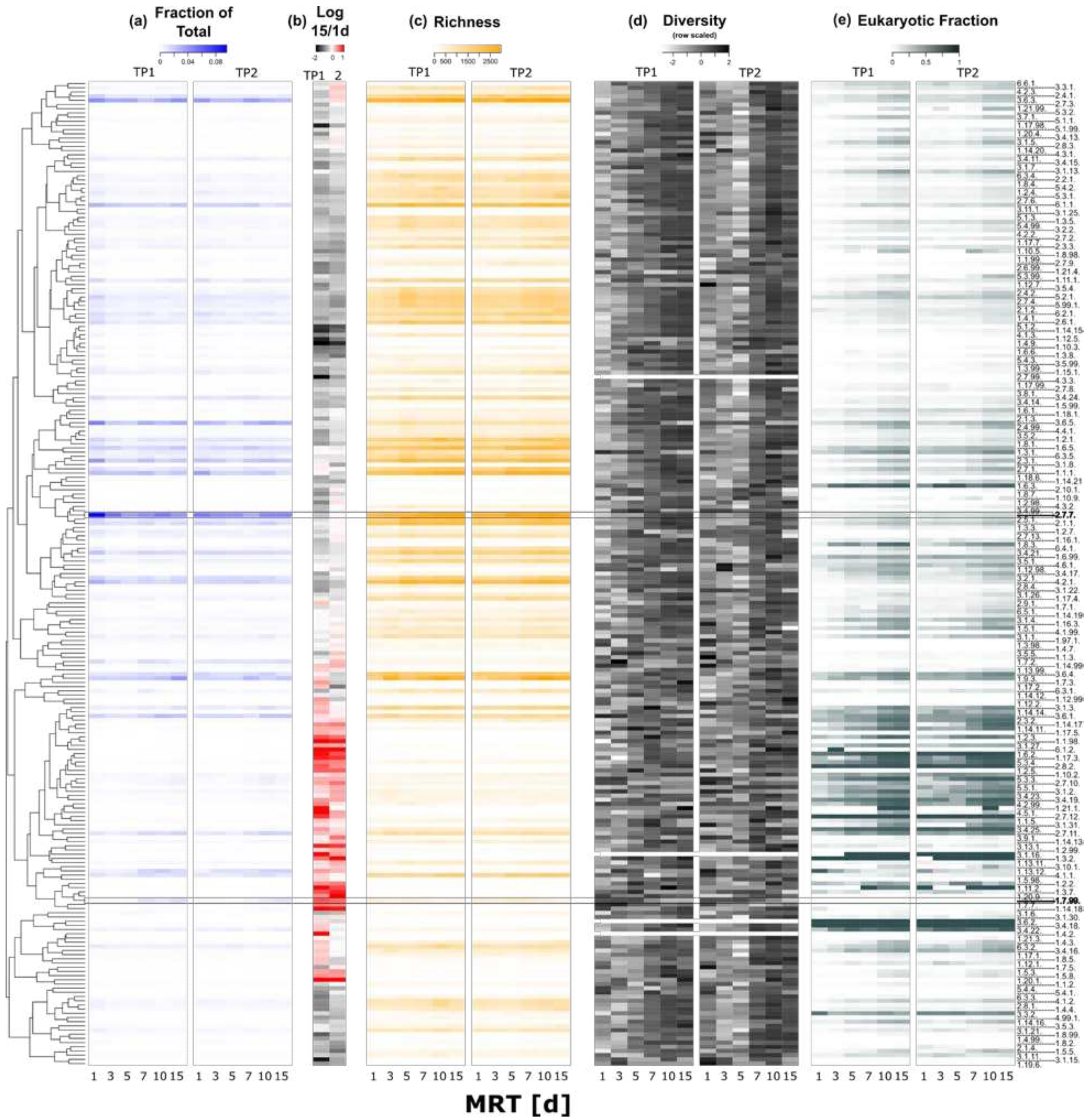
Supplemental Figure 14. The range of μ_{max} and K_s parameters for the 1 and 15 d MRT Sequencing Batch Reactor models when allowing only the μ_{max} and K_s parameters to vary (the b_e is held constant). For comparison, see Figure 1.b within the main text.



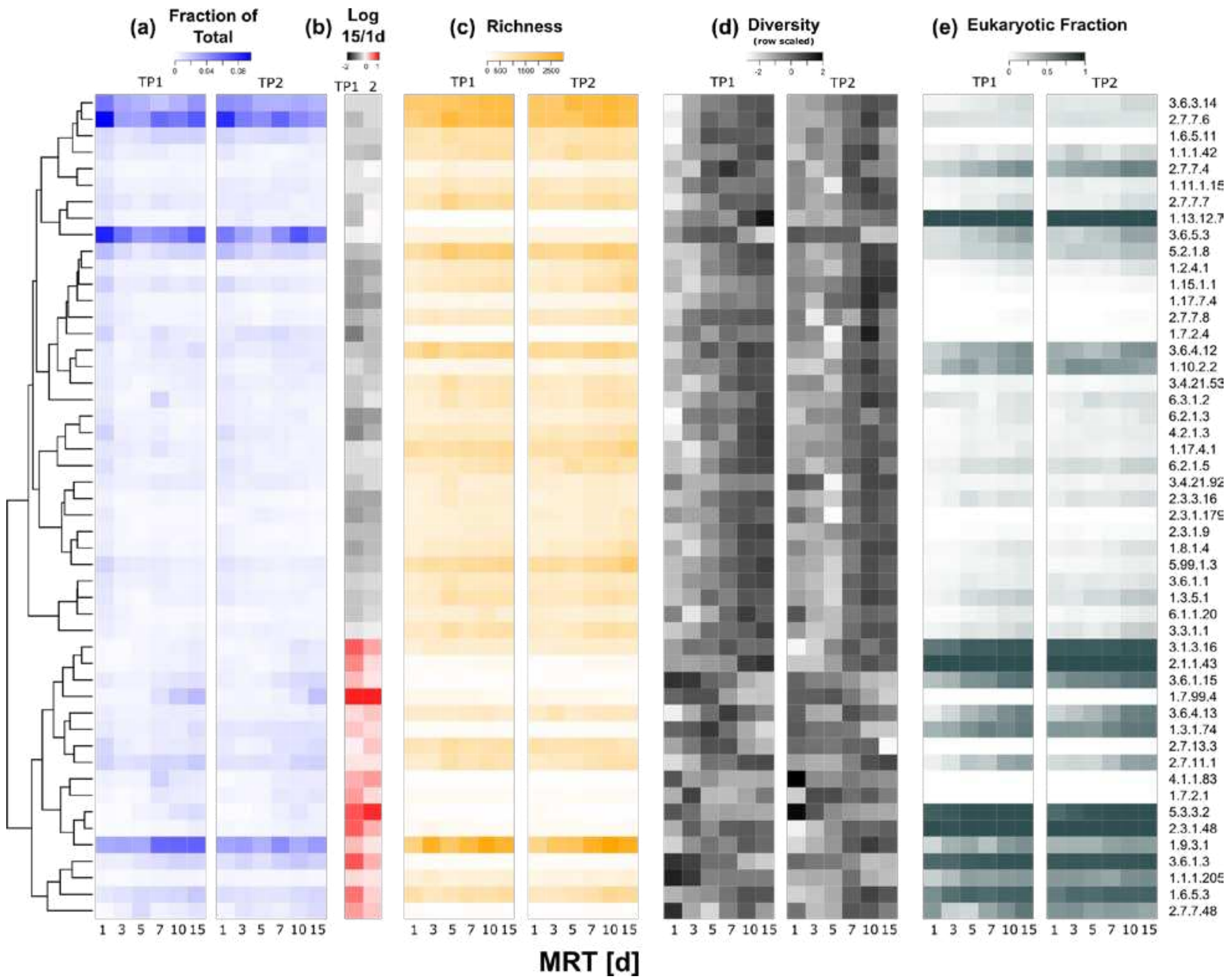
Supplemental Figure 15. A trace of the growth parameter solution path, the Shannon diversity, and the richness results of the model when the of μ_{max} and b_e are constrained (a) by the fastest grower or (b) by the slowest grower.



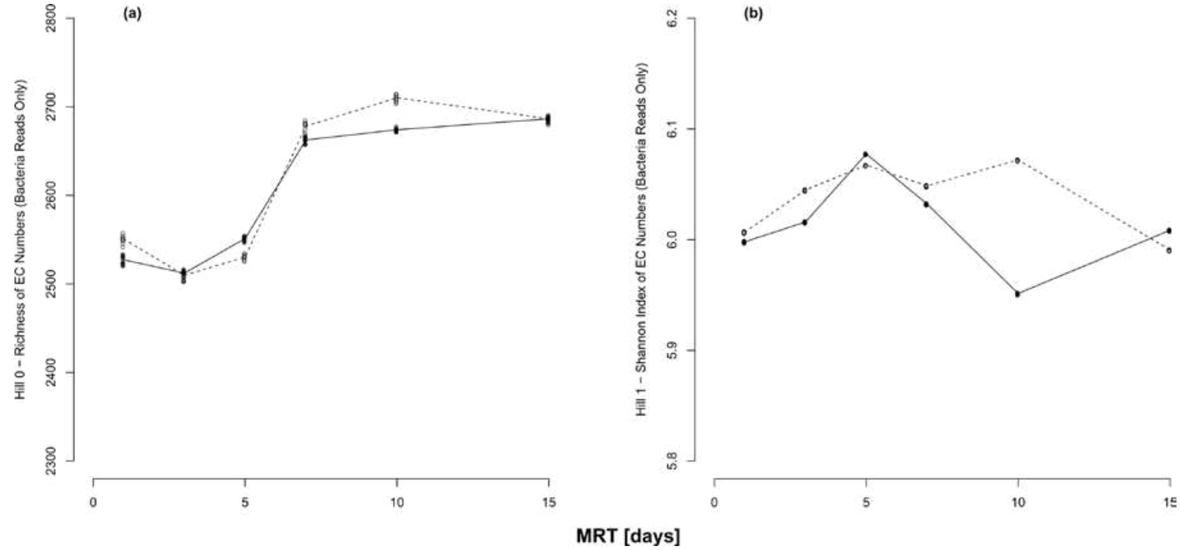
Supplemental Figure 16. a,b. The calculated richness and Shannon diversity for the B1-primer amplified 16S rRNA (black) and rDNA (light blue) data. c,d. The abundance data distributed into taxonomic orders; the top 10 of the sums across each experiment are colored, resulting in 12 orders being represented.



Supplementary Figure 17. Summary of all EC sub-subclasses across the MRT gradient. The heatmaps are organized hierarchically according to a Euclidean distance and Ward clustering of (a) the EC sub-subclass fraction of total abundance that is presented as the dendrogram. (b) The 15/1d MRT abundance log ratio. (c) Hill 0 richness and (d) Hill 1 diversity metrics were calculated based on the Uniport identifiers of the organism of origin that provided the annotation to the reads. (e) The fraction of the total reads that were annotated per EC sub-subclass originating from a Eukaryotic sequence in the Uniprot database.



Supplementary Figure 18. Summary of the top 50 maximum fractional abundances EC numbers across the MRT gradient. The heatmaps are organized hierarchically according to a Euclidean distance and Ward clustering of the (a) EC number fraction of total abundance that is presented as the dendrogram. (b) The 15/1d MRT abundance log ratio. The within EC number taxonomic (c) Hill 0 richness and (d) Hill 1 diversity metrics were calculated based on the Uniport identifiers of the organism of origin that provided the annotation to the reads. (e) The fraction of the total reads that were annotated per EC number originating from a Eukaryotic sequence in the Uniprot database.



Supplementary Figure 19. The richness $Hill_0$ (a) and Shannon Diversity $Hill_1$ (b) values for the functionally annotated RNA originating from Bacteria only data binned into common EC numbers for TP1 (solid) and TP2 (dashed) at the fractional abundance cutoff of 10^{-7}. The Hill 0 and 1 values were calculated for each reactor using the rtk package in R for 10 bootstrap sub-selections of the annotations. The lines trace the bootstrap mean.

Supplementary Files

Table of Contents

Supplemental File 1 : Phyloseq processing of the 16S rRNA reads

All commands detailed below were run using R v 3.5.1.

```
#Install required packages if not already available on the system
#=========================================================================================================
===
#source("https://bioconductor.org/biocLite.R")
#biocLite('DESeq2')
#biocLite('phyloseq')
#install.packages("ggfortify")
#install.packages("colorspace")
#install.packages("dplyr")
#install.packages("iNEXT")

#=========================================================================================================
===
#Load required libraries
#=========================================================================================================
===
library(phyloseq); packageVersion("phyloseq")     #At time of implementation, v 1.24.2
library(ggplot2); packageVersion("ggplot2")       #At time of implementation, v 2.2.1
library(reshape2); packageVersion("reshape2")     #At time of implementation, v 1.4.2
library(dplyr); packageVersion("dplyr")           #At time of implementation, v 0.7.1
library(grid); packageVersion("grid")             #At time of implementation, v 3.3.2
library(ggfortify); packageVersion("ggfortify")   #At time of implementation, v 0.4.1
library(gridExtra); packageVersion("gridExtra")   #At time of implementation, v 2.2.1
library(DESeq2); packageVersion("DESeq2")         #At time of implementation, v 1.14.1
library(biomformat); packageVersion("biomformat") #At time of implementation, v 1.2.0
library(vegan)
library(micorbiomeSeq)

#=========================================================================================================
===
# Import Files, these files are provided in the Supplmental Data .zip package
#=========================================================================================================
===

otufile_b1      <- "B1_ZOTU_Count_Sintax_SRT.txt"
mapfile_b1      <- "B1_metafile_SRT.txt"
treefile_b1     <- "B1_ZOTU_CLU.tre"
refseqfile_b1   <- "B1_ZOTU.fa"

otufile_b2      <- "B2_ZOTU_Count_Syntax_SRT.txt"
mapfile_b2      <- "B2_metafile_SRT.txt"
treefile_b2     <- "B2_ZOTU_CLU.tre"
refseqfile_b2   <- "B2_ZOTU.fa"

B1 <- import_qiime(otufilename = otufile_b1, mapfilename = mapfile_b1, treefilename = treefile_b1,
refseqfilename = refseqfile_b1)
B2 <- import_qiime(otufilename = otufile_b2, mapfilename = mapfile_b2, treefilename = treefile_b2,
refseqfilename = refseqfile_b2)

Exp1_B1 <- subset_samples(B1, Exp == "Exp1")
Exp1_B2 <- subset_samples(B2, Exp == "Exp1")
Exp2_B1 <- subset_samples(B1, Exp == "Exp2")
Exp2_B2 <- subset_samples(B2, Exp == "Exp2")

write.table(sample_sums(B1),"sample_sumsB1.txt")
#=========================================================================================================
===
# Counts of reads per sample
#=========================================================================================================
===
sort(sample_sums(Exp1_B1))
sort(sample_sums(Exp1_B2))
sort(sample_sums(Exp2_B1))
sort(sample_sums(Exp2_B2))
```

```
#==================================================================================================
===
# Plot abundance per sample to check for any low abundance remaining samples
#==================================================================================================
===
sumsE1B1 <- data.frame(sample_sums(Exp1_B1))
sumsE1B1$sampleID <- rownames(sumsE1B1)
sumsE1B1$sampleID <- c("1d cDNA 1","1d cDNA 2","1d cDNA 3","3d cDNA 1","3d cDNA 2","3d cDNA 3","5d cDNA
1","5d cDNA 2","5d cDNA 3",
                       "7d cDNA 1","7d cDNA 2","7d cDNA 3","10d cDNA 1","10d cDNA 2","10d cDNA 3","15d
cDNA 1","15d cDNA 2","15d cDNA 3",
                       "1d gDNA 1","1d gDNA 2","1d gDNA 3","3d gDNA 1","3d gDNA 2","3d gDNA 3","5d gDNA
1","5d gDNA 2","5d gDNA 3",
                       "7d gDNA 1","7d gDNA 2","7d gDNA 3","10d gDNA 1","10d gDNA 2","10d gDNA 3","15d
gDNA 1","15d gDNA 2","15d gDNA 3",
                       "[+] gDNA 1", "[+] gDNA 2", "[+] gDNA 3")
sumsE1B1$sampleID <- factor(sumsE1B1$sampleID, levels = sumsE1B1$sampleID)
E1B1bars <- ggplot(sumsE1B1, aes(sampleID,sample_sums.Exp1_B1.)) +
  geom_bar(stat="identity", width = 0.66) + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust
= 0.5)) +
  labs(y = "ESV Count", title = "TP1 B1")

sumsE1B2 <- data.frame(sample_sums(Exp1_B2))
sumsE1B2$sampleID <- rownames(sumsE1B2)
sumsE1B2$sampleID <- sumsE1B1$sampleID
sumsE1B2$sampleID <- factor(sumsE1B2$sampleID, levels = sumsE1B2$sampleID)
E1B2bars <- ggplot(sumsE1B2[sumsE1B2$sampleID != "[+] gDNA 3",], aes(sampleID,sample_sums.Exp1_B2.)) +
  geom_bar(stat="identity", width = 0.66) + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust
= 0.5)) +
  labs(y = "ESV Count", title = "TP1 B2")

sumsE2B1 <- data.frame(sample_sums(Exp2_B1))
sumsE2B1$sampleID <- rownames(sumsE2B1)
sumsE2B1$sampleID <- c("1d cDNA 1","1d cDNA 2","1d cDNA 3","3d cDNA 1","3d cDNA 2","3d cDNA 3","5d cDNA
1","5d cDNA 2","5d cDNA 3",
                       "7d cDNA 1","7d cDNA 2","7d cDNA 3","10d cDNA 1","10d cDNA 2","10d cDNA 3","15d
cDNA 1","15d cDNA 2","15d cDNA 3",
                       "1d gDNA 1","1d gDNA 2","1d gDNA 3","3d gDNA 1","3d gDNA 2","3d gDNA 3","5d gDNA
1","5d gDNA 2","5d gDNA 3",
                       "7d gDNA 1","7d gDNA 2","7d gDNA 3","10d gDNA 1","10d gDNA 2","10d gDNA 3","15d
gDNA 1","15d gDNA 2","15d gDNA 3",
                       "[-] gDNA")
sumsE2B1$sampleID <- factor(sumsE2B1$sampleID, levels = sumsE2B1$sampleID)
E2B1bars <- ggplot(sumsE2B1, aes(sampleID,sample_sums.Exp2_B1.)) +
  geom_bar(stat="identity", width = 0.66) + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust
= 0.5)) +
  labs(y = "ESV Count", title = "TP2 B1")

sumsE2B2 <- data.frame(sample_sums(Exp2_B2))
sumsE2B2$sampleID <- rownames(sumsE2B2)
sumsE2B2$sampleID <- c("1d cDNA 1","1d cDNA 2","1d cDNA 3","3d cDNA 1","3d cDNA 2","3d cDNA 3","5d cDNA
1","5d cDNA 2","5d cDNA 3",
                       "7d cDNA 1","7d cDNA 2","7d cDNA 3","10d cDNA 1","10d cDNA 2","10d cDNA 3","15d
cDNA 1","15d cDNA 2","15d cDNA 3",
                       "1d gDNA 1","1d gDNA 2","1d gDNA 3","3d gDNA 1","3d gDNA 2","3d gDNA 3","5d gDNA
1","5d gDNA 2","5d gDNA 3",
                       "7d gDNA 1","7d gDNA 2","7d gDNA 3","10d gDNA 1","10d gDNA 2","10d gDNA 3","15d
gDNA 1","15d gDNA 2","15d gDNA 3",
                       "[-] cDNA 1", "[-] cDNA 2", "[-] cDNA 3", "[+] gDNA 1", "[+] gDNA 2", "[-] gDNA 1",
"[-] gDNA 2")
sumsE2B2$sampleID <- factor(sumsE2B2$sampleID, levels = sumsE2B2$sampleID)
E2B2bars <- ggplot(sumsE2B2, aes(sampleID,sample_sums.Exp2_B2.)) +
  geom_bar(stat="identity", width = 0.66) + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust
= 0.5)) +
  labs(y = "ESV Count", title = "TP2 B2")

#abun_bar_e1_b1 <- plot_bar(Exp1_B1, title = "Exp1 Primer B1")
#abun_bar_e1_b2 <- plot_bar(Exp1_B2, title = "Exp1 Primer B2")
#abun_bar_e2_b1 <- plot_bar(Exp2_B1, title = "Exp2 Primer B1")
#abun_bar_e2_b2 <- plot_bar(Exp2_B2, title = "Exp2 Primer B2")
```

```
pdf("Supplemental Abundance BarPlot.pdf", height = 8.6, width = 10.7)
grid.arrange(E1B1bars, E2B1bars, E1B2bars, E2B2bars, ncol = 2)
dev.off()

# => One of the positive controls B2pos_3_Exp1_B2 has 10x reads and should be removed
# => B1gR4_2_Exp2_B1 does not have enough reads and should be removed

Exp1_B2 <- subset_samples(Exp1_B2, X.SampleID != "B2pos_3_Exp1_B2")
Exp2_B1 <- subset_samples(Exp2_B1, X.SampleID != "B1gR4_2_Exp2_B1")

#========================================================================================================
===
#Plot rarefaction curve per sample
#========================================================================================================
===

colscale <-
c("gold","gold","gold","orange","orange","orange","red","red","red","purple","purple","purple","blue","blu
e","blue","black","black","black",

"gold","gold","gold","orange","orange","orange","red","red","red","purple","purple","purple","blue","blue"
,"blue","black","black","black",
              "grey","grey","grey")
colscale2 <-
c("gold","gold","gold","orange","orange","orange","red","red","red","purple","purple","purple","blue","blu
e","blue","black","black","black",

"gold","gold","gold","orange","orange","orange","red","red","red","purple","purple","blue","blue","blue","
black","black","black",
              "grey","grey","grey")

ltyscale <- c(rep(1,18),rep(2,18),1,1,1)
ltyscale2 <- c(rep(1,18),rep(2,17),1,1,1)

pdf("Supplemental Rarefaction Curves.pdf", height = 8.6, width = 12.6)
par(mfrow=c(2,2),bty="n")
rarecurve(t(otu_table(Exp1_B1)), step=1000, lty=ltyscale, col=colscale, label=FALSE, xlab ="16S Reads Exp1
Primer B1", ylab = "ZOTU", ylim = c(0,4000), xlim = c(0,200000))
rarecurve(t(otu_table(Exp1_B2)), step=1000, lty=ltyscale, col=colscale, label=FALSE, xlab ="16S Reads Exp1
Primer B2", ylab = "ZOTU",  ylim = c(0,4000), xlim = c(0,200000))
rarecurve(t(otu_table(Exp2_B1)), step=1000, lty=ltyscale2, col=colscale2, label=FALSE, xlab ="16S Reads
Exp2 Primer B1", ylab = "ZOTU",  ylim = c(0,4000), xlim = c(0,200000))
rarecurve(t(otu_table(Exp2_B2)), step=1000, lty=c(ltyscale,1,1,1), col=c(colscale,rep("grey",6)), ylab =
"ZOTU", label=FALSE, xlab ="16S Reads Exp2 Primer B2", ylim = c(0,4000), xlim = c(0,200000))
dev.off()
dev.off()

#========================================================================================================
===
# Rarefy data for each B1 and B2 primer combinations seperately.
#========================================================================================================
===
Exp1_B1_even = rarefy_even_depth(subset_samples(Exp1_B1, SludgeType == "SRT"), rngseed = 1013)
Exp1_B2_even = rarefy_even_depth(subset_samples(Exp1_B2, SludgeType == "SRT"), rngseed = 1013)
Exp2_B1_even = rarefy_even_depth(subset_samples(Exp2_B1, SludgeType == "SRT"), rngseed = 1013)
Exp2_B2_even = rarefy_even_depth(subset_samples(Exp2_B2, SludgeType == "SRT"), rngseed = 1013)

sample_sums(Exp1_B1_even)
#========================================================================================================
===
#========================================================================================================
===
# Produce final summary graphic, run in text analyses
#========================================================================================================
===

# Set the colors of the top orders
#===============================================
order_colors = rep(c(
```

```
  Bdellovibrionales='gray2',
  Burkholderiales='gray30',
  Caldilineales = 'indianred3',
  Flavobacteriales='gray50',
  Lactobacillales='red3',
  Micrococcales = 'firebrick1',
  Myxococcales='lightskyblue3',
  Nitrosomonadales='gold1',
  Planctomycetales='rosybrown1',
  Pseudomonadales='dodgerblue4',
  Rhodobacterales='royalblue1',
  Rhodocyclales='orange3',
  Sphingobacteriales='lightgoldenrod2',
  Thiotrichales='darkorange4',
  Verrucomicrobiales='navajowhite4'
), 20)
#===============================================
# Aggregate samples into orders, plot abundance bars
#===============================================
glomExp1_B1_even_c <- tax_glom(subset_samples(Exp1_B1_even, Type %in% c("16S rRNA","16S rDNA")),
taxrank='Order')
div_by_rep_for_mean_rollup <- otu_table(glomExp1_B1_even_c)
otu_table(glomExp1_B1_even_c) <- div_by_rep_for_mean_rollup/c(rep(3,36))
gE1B1ec_filt = filter_taxa(glomExp1_B1_even_c, function(x) max(x) > 0, TRUE)
glomExp1_B1_even_c.long <- psmelt(gE1B1ec_filt)
glomExp1_B1_even_c.long <- arrange(glomExp1_B1_even_c.long, Order)

write.table(glomExp1_B1_even_c.long,"longcheck.txt")

bar_e1_b1 <- ggplot(glomExp1_B1_even_c.long, aes(x = ST_RN, y = Abundance, fill = Order)) +
facet_grid(Type~.) +
  geom_bar(stat="summary", position="stack", fun.y = "sum") + theme(legend.text=element_text(size=6)) +
ggtitle("c. Exp1 Taxonomic Orders") +
  scale_fill_manual(values = order_colors, na.value = 'white') +
  theme(legend.position="none") +
  scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" = "7","SRT5" = "10","SRT6" =
"15")) +
  theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x = element_blank(), axis.title.y =
element_blank(), strip.background  = element_blank(),
                      axis.line = element_blank(), panel.grid.major = element_line(colour="white"),
panel.grid.minor = element_line(colour="white"),
                      panel.background = element_blank(), panel.border = element_blank())

orders10_exp1 <- names(sort(taxa_sums(subset_samples(glomExp1_B1_even_c, Type == "16S rRNA")), TRUE))
taxon10_exp1  <- tax_table(glomExp1_B1_even_c)
otu10_exp1 <- otu_table(glomExp1_B1_even_c)
write.table(cbind(otu10_exp1,taxon10_exp1),"InvestigationofexpirRNAotu.txt")
genus_exp1  <-taxon10_exp1 [orders10_exp1 ]

glomExp1_B2_even_c <- tax_glom(subset_samples(Exp1_B2_even, Type %in% c("16S rRNA","16S rDNA")),
taxrank='Order')
div_by_rep_for_mean_rollup <- otu_table(glomExp1_B2_even_c)
otu_table(glomExp1_B2_even_c) <- div_by_rep_for_mean_rollup/c(rep(3,36))
gE1B2ec_filt = filter_taxa(glomExp1_B2_even_c, function(x) max(x) > 0, TRUE)
glomExp1_B2_even_c.long <- psmelt(gE1B2ec_filt)
glomExp1_B2_even_c.long <- arrange(glomExp1_B2_even_c.long, Order)

orders10_exp1 <- names(sort(taxa_sums(glomExp1_B2_even_c), TRUE)[1:50])
sample_sums(glomExp1_B2_even_c) #83962
taxon10_exp1  <- tax_table(glomExp1_B2_even_c)
genus_exp1  <-taxon10_exp1 [orders10_exp1 ]
otu_count_exp1_RNA  <- otu_table(subset_samples(glomExp1_B2_even_c, Type == "16S rRNA"))
otu_count_exp1_DNA  <- otu_table(subset_samples(glomExp1_B2_even_c, Type == "16S rDNA"))

RNA_orders_exp1 <- otu_count_exp1_RNA[names(sort(taxa_sums(glomExp1_B2_even_c), TRUE)[1:50])]
DNA_orders_exp1 <- otu_count_exp1_DNA[names(sort(taxa_sums(glomExp1_B2_even_c), TRUE)[1:50])]

cor(RNA_orders_exp1,DNA_orders_exp1, method = "spearman")
```

```
bar_e1_b2 <- ggplot(glomExp1_B2_even_c.long, aes(x = ST_RN, y = Abundance, fill = Order)) +
facet_grid(Type~.) +
  geom_bar(stat="summary", position="stack", fun.y = "sum") + theme(legend.text=element_text(size=6)) +
ggtitle("c. Exp1 Taxonomic Orders") +
  scale_fill_manual(values = order_colors, na.value = 'white') +
  theme(legend.position="none") +
  scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" = "7","SRT5" = "10","SRT6" =
"15")) +
  theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x = element_blank(), axis.title.y =
element_blank(), strip.background  = element_blank(),
        axis.line = element_blank(), panel.grid.major = element_line(colour="white"), panel.grid.minor =
element_line(colour="white"),
        panel.background = element_blank(), panel.border = element_blank())

glomExp2_B1_even_c <- tax_glom(subset_samples(Exp2_B1_even, Type %in% c("16S rRNA","16S rDNA")),
taxrank='Order')
div_by_rep_for_mean_rollup <- otu_table(glomExp2_B1_even_c)
div_by_rep_for_mean_rollup[,28:29] <- div_by_rep_for_mean_rollup[,28:29]*1.5
otu_table(glomExp2_B1_even_c) <- div_by_rep_for_mean_rollup/3
gE2B1ec_filt = filter_taxa(glomExp2_B1_even_c, function(x) max(x) > 0, TRUE)
glomExp2_B1_even_c.long <- psmelt(gE2B1ec_filt)
glomExp2_B1_even_c.long <- arrange(glomExp2_B1_even_c.long, Order)

bar_e2_b1 <- ggplot(glomExp2_B1_even_c.long, aes(x = ST_RN, y = Abundance, fill = Order)) +
facet_grid(Type~.) +
  geom_bar(stat="summary", position="stack", fun.y = "sum") + theme(legend.text=element_text(size=6)) +
ggtitle("d. Exp 2") +
  scale_fill_manual(values = order_colors, na.value = "white")  + theme(legend.position="none") +
  scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" = "7","SRT5" = "10","SRT6" =
"15")) +
  theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x = element_blank(), axis.title.y =
element_blank(), strip.background  = element_blank(),
        axis.line = element_blank(), panel.grid.major = element_line(colour="white"), panel.grid.minor =
element_line(colour="white"),
        panel.background = element_blank(), panel.border = element_blank())

glomExp2_B2_even_c <- tax_glom(subset_samples(Exp2_B2_even, Type %in% c("16S rRNA","16S rDNA")),
taxrank='Order')
div_by_rep_for_mean_rollup <- otu_table(glomExp2_B2_even_c)
otu_table(glomExp2_B2_even_c) <- div_by_rep_for_mean_rollup/3
gE2B2ec_filt = filter_taxa(glomExp2_B2_even_c, function(x) max(x) > 0, TRUE)
glomExp2_B2_even_c.long <- psmelt(gE2B2ec_filt)
glomExp2_B2_even_c.long <- arrange(glomExp2_B2_even_c.long, Order)

orders10_exp2 <- names(sort(taxa_sums(glomExp2_B2_even_c), TRUE)[1:20])
sample_sums(glomExp2_B2_even_c) #83962
taxon10_exp2  <- tax_table(glomExp2_B2_even_c)
genus_exp2  <-taxon10_exp2[orders10_exp2]
otu_count_exp2  <- otu_table(glomExp2_B2_even_c)

otu_count_exp2_RNA  <- otu_table(subset_samples(glomExp2_B2_even_c, Type == "16S rRNA"))
otu_count_exp2_DNA  <- otu_table(subset_samples(glomExp2_B2_even_c, Type == "16S rDNA"))

RNA_orders_exp2 <- otu_count_exp2_RNA[names(sort(taxa_sums(glomExp2_B2_even_c), TRUE)[1:50])]
DNA_orders_exp2 <- otu_count_exp2_DNA[names(sort(taxa_sums(glomExp2_B2_even_c), TRUE)[1:50])]

write.table(cor(RNA_orders_exp2,DNA_orders_exp2, method = "spearman"), "exp2ordercorrelation.txt")
write.table(cor(RNA_orders_exp1,DNA_orders_exp2, method = "spearman"), "exp1ordercorrelation.txt")


bar_e2_b2 <- ggplot(glomExp2_B2_even_c.long, aes(x = ST_RN, y = Abundance, fill = Order)) +
facet_grid(Type~.) +
  geom_bar(stat="summary", position="stack", fun.y = "sum") + theme(legend.text=element_text(size=6)) +
ggtitle("d. Exp 2") +
  scale_fill_manual(values = order_colors, na.value = "white")  + theme(legend.position="none") +
  scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" = "7","SRT5" = "10","SRT6" =
"15")) +
  theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x = element_blank(), axis.title.y =
element_blank(), strip.background  = element_blank(),
```

```
        axis.line = element_blank(), panel.grid.major = element_line(colour="white"), panel.grid.minor =
element_line(colour="white"),
        panel.background = element_blank(), panel.border = element_blank())

#===============================================
# In text analysis: Comparison of 15d to 1d for Burkholderiales, Rhodocyclales, and Myxococcales
#===============================================
burk_count_exp1 <- otu_count_exp1 ["ZOTU28",]/83962*100
burk_analysis_exp1 <-
c(mean(burk_count_exp1[1,1:3]),sd(burk_count_exp1[1,1:3]),mean(burk_count_exp1[1,16:18]),sd(burk_count_exp
1[1,16:18]))

rhodo_count_exp1 <- otu_count_exp1["ZOTU2",]/54554*100
rhodo_analysis_exp1 <-
c(mean(rhodo_count_exp1[1,1:3]),sd(rhodo_count_exp1[1,1:3]),mean(rhodo_count_exp1[1,16:18]),sd(rhodo_count
_exp1[1,16:18]))

myxo_count_exp1 <- otu_count_exp1["ZOTU284",]/54554*100
myxo_analysis_exp1 <-
c(mean(myxo_count_exp1[1,1:3]),sd(myxo_count_exp1[1,1:3]),mean(myxo_count_exp1[1,16:18]),sd(myxo_count_exp
1[1,16:18]))

burk_count_exp2 <- otu_count_exp2["ZOTU10",]/54554*100
burk_analysis_exp2 <-
c(mean(burk_count_exp2[1,1:3]),sd(burk_count_exp2[1,1:3]),mean(burk_count_exp2[1,16:18]),sd(burk_count_exp
2[1,16:18]))

rhodo_count_exp2 <- otu_count_exp2["ZOTU2",]/54554*100
rhodo_analysis_exp2 <-
c(mean(rhodo_count_exp2[1,1:3]),sd(rhodo_count_exp2[1,1:3]),mean(rhodo_count_exp2[1,16:18]),sd(rhodo_count
_exp2[1,16:18]))

myxo_count_exp2 <- otu_count_exp2["ZOTU20",]/54554*100
myxo_analysis_exp2 <-
c(mean(myxo_count_exp2[1,1:3]),sd(myxo_count_exp2[1,1:3]),mean(myxo_count_exp2[1,16:18]),sd(myxo_count_exp
2[1,16:18]))

#===============================================
# Create appropriate labels as grobs.
#===============================================

planc <- textGrob("Planctomycetales",gp=gpar(col="rosybrown1", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
cald <- textGrob("Caldilineales",gp=gpar(col="indianred3", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
micro <- textGrob("Micrococcales",gp=gpar(col="firebrick1", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
bdel <- textGrob("Bdellovibrionales",gp=gpar(col="grey2", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
burk <- textGrob("Burkholderiales",gp=gpar(col="gray30", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
flav <- textGrob("Flavobacteriales",gp=gpar(col="gray50", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
myx <- textGrob("Myxococcales",gp=gpar(col="lightskyblue3", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
nitro <- textGrob("Nitrosomonadales",gp=gpar(col="gold1", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
lacto <- textGrob("Lactobacillales",gp=gpar(col="red3", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
sphingo <- textGrob("Sphingobacteriales",gp=gpar(col="lightgoldenrod2", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
rhodo <- textGrob("Rhodocyclales",gp=gpar(col="orange3", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
rhodob <- textGrob("Rhodobacterales",gp=gpar(col="royalblue1", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
psd <- textGrob("Pseudomonadales",gp=gpar(col="dodgerblue4", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
thio <- textGrob("Thiotrichales",gp=gpar(col="darkorange4", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
```

```
ver <- textGrob("Verrucomicrobiales",gp=gpar(col="navajowhite4", fontsize=10,
fontface="bold.italic"),x=0.1,just="left")
all_else <- textGrob("Other",gp=gpar(col="white", fontsize=10, fontface="bold.italic"),x=0.1,just="left")

theme_set(theme_bw())

blank <- textGrob("",gp=gpar(col="black"))
leg_cDNA <- textGrob(". 16S rRNA",gp=gpar(col="black", fontsize=10, fontface="bold"),x=0.1,just="left")
leg_gDNA <- textGrob(". 16S rDNA",gp=gpar(col="light blue", fontsize=10,
fontface="bold"),x=0.1,just="left")
title_exp1 <- textGrob("Exp1 48d",gp=gpar(col="black", fontsize=16, fontface="bold"),x=0.1,just="left")
title_exp2 <- textGrob("Exp2 187d",gp=gpar(col="black", fontsize=16, fontface="bold"),x=0.1,just="left")
xaxis_lab <- textGrob("MRT [d]",gp=gpar(col="black", fontsize=12, fontface="bold"),x=0.1,just="left")

#===============================================
# Plot the richness versus MRT
#===============================================
Exp1_B1_richness_obs  <- plot_richness(Exp1_B1_even, "ST_RN", "Type", title = "a. Richness",
measures=c("Observed")) + scale_colour_manual(values = c("light blue", "black")) +
theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" =
"7","SRT5" = "10","SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x =
element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(1000,2500), breaks =
c(1000, 1500, 2000, 2500)) + theme( strip.background = element_blank(),  strip.text.x = element_blank(),
panel.border = element_blank(), panel.background = element_blank())

Exp1_B2_richness_obs  <- plot_richness(Exp1_B2_even, "ST_RN", "Type", title = "a. Richness",
measures=c("Observed")) + scale_colour_manual(values = c("light blue", "black")) +
theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" =
"7","SRT5" = "10","SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x =
element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(1000,2500), breaks =
c(1000, 1500, 2000, 2500)) + theme( strip.background = element_blank(),  strip.text.x = element_blank(),
panel.border = element_blank(), panel.background = element_blank())

Exp2_B1_richness_obs  <- plot_richness(Exp2_B1_even, "ST_RN", "Type", title = "b. Richness",
measures=c("Observed")) + scale_colour_manual(values = c("light blue", "black")) +
theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" =
"7","SRT5" = "10","SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x =
element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(1000,2500), breaks =
c(1000, 1500, 2000, 2500)) + theme( strip.background = element_blank(),  strip.text.x = element_blank(),
panel.border = element_blank(), panel.background = element_blank())

Exp2_B2_richness_obs  <- plot_richness(Exp2_B2_even, "ST_RN", "Type", title = "b. Richness",
measures=c("Observed")) + scale_colour_manual(values = c("light blue", "black")) +
theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" = "5","SRT4" =
"7","SRT5" = "10","SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0), axis.title.x =
element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(1000,2500), breaks =
c(1000, 1500, 2000, 2500)) + theme( strip.background = element_blank(),  strip.text.x = element_blank(),
panel.border = element_blank(), panel.background = element_blank())

rich <- rbind(Exp1_B2_richness_obs$data,Exp2_B2_richness_obs$data)
write.table(rich,"ASVrich.txt")

#===============================================
# Plot the Shannon Diversity vs MRT
# Note : must consider raw data.
#===============================================
Exp1_B1_richness_sha  <- plot_richness(subset_samples(Exp1_B1, SludgeType == "SRT"), "ST_RN", "Type",
title = "Shannon Diversity", measures=c("Shannon")) + scale_colour_manual(values = c("light blue",
"black")) + theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" =
"5","SRT4" = "7","SRT5" = "10","SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0),
axis.title.x = element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(2,6),
breaks = c(2, 3, 4, 5, 6)) + theme( strip.background = element_blank(),  strip.text.x = element_blank(),
panel.border = element_blank(), panel.background = element_blank())
Exp1_B2_richness_sha  <- plot_richness(subset_samples(Exp1_B2, SludgeType == "SRT"), "ST_RN", "Type",
title = "Shannon Diversity", measures=c("Shannon")) + scale_colour_manual(values = c("light blue",
"black")) + theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" =
"5","SRT4" = "7","SRT5" = "10","SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0),
axis.title.x = element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(2,6),
breaks = c(2, 3, 4, 5, 6)) + theme( strip.background = element_blank(),  strip.text.x = element_blank(),
panel.border = element_blank(), panel.background = element_blank())
```

```
Exp2_B1_richness_sha  <- plot_richness(subset_samples(Exp2_B1, SludgeType == "SRT"), "ST_RN", "Type",
title = "Shannon Diversity", measures=c("Shannon")) + scale_colour_manual(values = c("light blue",
"black")) + theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" =
"5","SRT4" = "7","SRT5" = "10","SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0),
axis.title.x = element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(2,6),
breaks = c(2, 3, 4, 5, 6)) + theme( strip.background = element_blank(),  strip.text.x = element_blank(),
panel.border = element_blank(), panel.background = element_blank())
Exp2_B2_richness_sha  <- plot_richness(subset_samples(Exp2_B2_even, SludgeType == "SRT"), "ST_RN", "Type",
title = "Shannon Diversity", measures=c("Shannon")) + scale_colour_manual(values = c("light blue",
"black")) + theme(legend.position="none") + scale_x_discrete(labels=c("SRT1" = "1", "SRT2" = "3","SRT3" =
"5","SRT4" = "7","SRT5" = "10","SRT6" = "15")) + theme(axis.text.x = element_text(face="bold", angle=0),
axis.title.x = element_blank(), axis.title.y = element_blank()) + scale_y_continuous(limits = c(2,6),
breaks = c(2, 3, 4, 5, 6)) + theme( strip.background = element_blank(),  strip.text.x = element_blank(),
panel.border = element_blank(), panel.background = element_blank())

#===============================================
# Run all correlation comparisons
#===============================================

shan <- rbind(Exp1_B2_richness_sha$data,Exp2_B2_richness_sha$data)
write.table(shan,"ASVshan.txt")
diversity_sum <- data.frame(rich$X.SampleID,rich$Type,rich$Exp,rich$ReactorNum,rich$value,shan$value)

exp1_rRNA_div <- diversity_sum[which(diversity_sum$rich.Type=='16S rRNA' & diversity_sum$rich.Exp ==
"Exp1"),]
exp2_rRNA_div <- diversity_sum[which(diversity_sum$rich.Type=='16S rRNA' & diversity_sum$rich.Exp ==
"Exp2"),]
exp1_rDNA_div <- diversity_sum[which(diversity_sum$rich.Type=='16S rDNA' & diversity_sum$rich.Exp ==
"Exp1"),]
exp2_rDNA_div <- diversity_sum[which(diversity_sum$rich.Type=='16S rDNA' & diversity_sum$rich.Exp ==
"Exp2"),]

e1_rich_p <- cor(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp1_rDNA_div,
tapply(rich.value, rich.ReactorNum, mean)))
e1_rich_s <- cor(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp1_rDNA_div,
tapply(rich.value, rich.ReactorNum, mean)), method="spearman")
e1_rich_l <- sqrt(summary(lm(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum,
mean))~with(exp1_rDNA_div, tapply(rich.value, rich.ReactorNum, mean))))$adj.r.squared)
e1_shan_p <- cor(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp1_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean)))
e1_shan_s <- cor(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp1_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean)), method="spearman")
e1_shan_l <- summary(lm(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean))~with(exp1_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean))))$adj.r.squared

e2_rich_p <- cor(with(exp2_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
tapply(rich.value, rich.ReactorNum, mean)))
e2_rich_s <- cor(with(exp2_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
tapply(rich.value, rich.ReactorNum, mean)), method="spearman")
e2_rich_l <- sqrt(summary(lm(with(exp2_rRNA_div, tapply(rich.value, rich.ReactorNum,
mean))~with(exp2_rDNA_div, tapply(rich.value, rich.ReactorNum, mean))))$adj.r.squared)
e2_shan_p <- cor(with(exp2_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean)))
e2_shan_s <- cor(with(exp2_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean)), method="spearman")
e2_shan_l <- summary(lm(with(exp2_rRNA_div, tapply(shan.value, rich.ReactorNum, mean))~with(exp2_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean))))$adj.r.squared

DNA_RNA_pearson_summary <- data.frame(e1_rich_p,e1_shan_p, e2_rich_p, e2_shan_p)
DNA_RNA_spearman_summary <- data.frame(e1_rich_s,e1_shan_s, e2_rich_s, e2_shan_s)

e1_rich_p <- cor(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp2_rRNA_div,
tapply(rich.value, rich.ReactorNum, mean)))
e1_rich_s <- cor(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp2_rRNA_div,
tapply(rich.value, rich.ReactorNum, mean)), method="spearman")
e1_rich_l <- sqrt(summary(lm(with(exp1_rRNA_div, tapply(rich.value, rich.ReactorNum,
mean))~with(exp2_rRNA_div, tapply(rich.value, rich.ReactorNum, mean))))$adj.r.squared)
e1_shan_p <- cor(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rRNA_div,
tapply(shan.value, rich.ReactorNum, mean)))
```

```
e1_shan_s <- cor(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rRNA_div,
tapply(shan.value, rich.ReactorNum, mean)), method="spearman")
e1_shan_l <- summary(lm(with(exp1_rRNA_div, tapply(shan.value, rich.ReactorNum, mean))~with(exp2_rRNA_div,
tapply(shan.value, rich.ReactorNum, mean))))$adj.r.squared

e1_rich_p <- cor(with(exp1_rDNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
tapply(rich.value, rich.ReactorNum, mean)))
e1_rich_s <- cor(with(exp1_rDNA_div, tapply(rich.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
tapply(rich.value, rich.ReactorNum, mean)), method="spearman")
e1_rich_l <- sqrt(summary(lm(with(exp1_rDNA_div, tapply(rich.value, rich.ReactorNum,
mean))~with(exp2_rDNA_div, tapply(rich.value, rich.ReactorNum, mean))))$adj.r.squared)
e1_shan_p <- cor(with(exp1_rDNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean)))
e1_shan_s <- cor(with(exp1_rDNA_div, tapply(shan.value, rich.ReactorNum, mean)),with(exp2_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean)), method="spearman")
e1_shan_l <- summary(lm(with(exp1_rDNA_div, tapply(shan.value, rich.ReactorNum, mean))~with(exp2_rDNA_div,
tapply(shan.value, rich.ReactorNum, mean))))$adj.r.squared

DNA_RNA_pearson_summary <- data.frame(e1_rich_p,e1_shan_p, e2_rich_p, e2_shan_p)
DNA_RNA_spearman_summary <- data.frame(e1_rich_s,e1_shan_s, e2_rich_s, e2_shan_s)

MRTe1_rich_p <- cor(exp1_rRNA_div$rich.value,exp1_rDNA_div$rich.ReactorNum)
MRTe1_rich_s <- cor(exp1_rRNA_div$rich.value,exp1_rDNA_div$rich.ReactorNum, method = "spearman")
MRTe1_rich_l <- sqrt(summary(lm(exp1_rRNA_div$rich.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
MRTe1_shan_p <- cor(exp1_rRNA_div$shan.value,exp1_rDNA_div$rich.ReactorNum)
MRTe1_shan_s <- cor(exp1_rRNA_div$shan.value,exp1_rDNA_div$rich.ReactorNum, method = "spearman")
MRTe1_shan_l <- sqrt(summary(lm(exp1_rRNA_div$shan.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)

MRTe2_rich_p <- cor(exp2_rRNA_div$rich.value,exp2_rDNA_div$rich.ReactorNum)
MRTe2_rich_s <- cor(exp2_rRNA_div$rich.value,exp2_rDNA_div$rich.ReactorNum, method = "spearman")
MRTe2_rich_l <- sqrt(summary(lm(exp2_rRNA_div$rich.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
MRTe2_shan_p <- cor(exp2_rRNA_div$shan.value,exp2_rDNA_div$rich.ReactorNum)
MRTe2_shan_s <- cor(exp2_rRNA_div$shan.value,exp2_rDNA_div$rich.ReactorNum, method = "spearman")
MRTe2_shan_l <- sqrt(summary(lm(exp2_rRNA_div$shan.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)

MRT_rRNA_pearson_summary <- data.frame(MRTe1_rich_p,MRTe1_shan_p, MRTe2_rich_p, MRTe2_shan_p)
MRT_rRNA_spearman_summary <- data.frame(MRTe1_rich_s, MRTe1_shan_s, MRTe2_rich_s, MRTe2_shan_s)

MRTe1_rich_p <- cor(exp1_rDNA_div$rich.value,exp1_rDNA_div$rich.ReactorNum)
MRTe1_rich_s <- cor(exp1_rDNA_div$rich.value,exp1_rDNA_div$rich.ReactorNum, method = "spearman")
MRTe1_rich_l <- sqrt(summary(lm(exp1_rDNA_div$rich.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
MRTe1_shan_p <- cor(exp1_rDNA_div$shan.value,exp1_rDNA_div$rich.ReactorNum)
MRTe1_shan_s <- cor(exp1_rDNA_div$shan.value,exp1_rDNA_div$rich.ReactorNum, method = "spearman")
MRTe1_shan_l <- sqrt(summary(lm(exp1_rDNA_div$shan.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)

MRTe2_rich_p <- cor(exp2_rDNA_div$rich.value,exp2_rDNA_div$rich.ReactorNum)
MRTe2_rich_s <- cor(exp2_rDNA_div$rich.value,exp2_rDNA_div$rich.ReactorNum, method = "spearman")
MRTe2_rich_l <- sqrt(summary(lm(exp2_rDNA_div$rich.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)
MRTe2_shan_p <- cor(exp2_rDNA_div$shan.value,exp2_rDNA_div$rich.ReactorNum)
MRTe2_shan_s <- cor(exp2_rDNA_div$shan.value,exp2_rDNA_div$rich.ReactorNum, method = "spearman")
MRTe2_shan_l <- sqrt(summary(lm(exp2_rDNA_div$shan.value~exp2_rDNA_div$rich.ReactorNum))$adj.r.squared)

MRT_rDNA_pearson_summary <- data.frame(MRTe1_rich_p,MRTe1_shan_p, MRTe2_rich_p, MRTe2_shan_p)
MRT_rDNA_spearman_summary <- data.frame(MRTe1_rich_s, MRTe1_shan_s, MRTe2_rich_s, MRTe2_shan_s)

mRNA_exp1_rich <- rep(c(2949.8, 2968.3, 3068.4, 3214.5, 3348.0, 3362.7), each=3)
mRNA_exp2_rich <- rep(c(3001.3, 2994.5, 3034.3, 3322.0, 3382.9, 3350.3), each=3)
mRNA_exp1_shan <- rep(c(5.9916,6.01818,6.08388,6.06815,5.9818,6.00447), each=3)
mRNA_exp2_shan <- rep(c(6.00442,6.01225,6.08577,6.05381,6.01788,6.0194), each=3)

rDNA_mRNA_e1_rich_p <- cor(exp1_rDNA_div$rich.value,mRNA_exp1_rich)
rDNA_mRNA_e1_rich_s <- cor(exp1_rDNA_div$rich.value,mRNA_exp1_rich, method = "spearman")
rDNA_mRNA_e1_rich_l <- sqrt(summary(lm(exp1_rDNA_div$rich.value~mRNA_exp1_rich))$adj.r.squared)

rRNA_mRNA_e1_rich_p <- cor(exp1_rRNA_div$rich.value,mRNA_exp1_rich)
rRNA_mRNA_e1_rich_s <- cor(exp1_rRNA_div$rich.value,mRNA_exp1_rich, method = "spearman")
rRNA_mRNA_e1_rich_l <- sqrt(summary(lm(exp1_rRNA_div$rich.value~mRNA_exp1_rich))$adj.r.squared)

rDNA_mRNA_e2_rich_p <- cor(exp2_rDNA_div$rich.value,mRNA_exp2_rich)
rDNA_mRNA_e2_rich_s <- cor(exp2_rDNA_div$rich.value,mRNA_exp2_rich, method = "spearman")
```

```
rDNA_mRNA_e2_rich_l <- sqrt(summary(lm(exp2_rDNA_div$rich.value~mRNA_exp2_rich))$adj.r.squared)

rRNA_mRNA_e2_rich_p <- cor(exp2_rRNA_div$rich.value,mRNA_exp2_rich)
rRNA_mRNA_e2_rich_s <- cor(exp2_rRNA_div$rich.value,mRNA_exp2_rich, method = "spearman")
rRNA_mRNA_e2_rich_l <- sqrt(summary(lm(exp2_rRNA_div$rich.value~mRNA_exp2_rich))$adj.r.squared)

rDNA_mRNA_e1_shan_p <- cor(exp1_rDNA_div$shan.value,mRNA_exp1_shan)
rDNA_mRNA_e1_shan_s <- cor(exp1_rDNA_div$shan.value,mRNA_exp1_shan, method = "spearman")
rDNA_mRNA_e1_shan_l <- sqrt(summary(lm(exp1_rDNA_div$shan.value~mRNA_exp1_shan))$adj.r.squared)

rRNA_mRNA_e1_shan_p <- cor(exp1_rRNA_div$shan.value,mRNA_exp1_shan)
rRNA_mRNA_e1_shan_s <- cor(exp1_rRNA_div$shan.value,mRNA_exp1_shan, method = "spearman")
rRNA_mRNA_e1_shan_l <- sqrt(summary(lm(exp1_rRNA_div$shan.value~mRNA_exp1_shan))$adj.r.squared)

rDNA_mRNA_e2_shan_p <- cor(exp2_rDNA_div$shan.value,mRNA_exp2_shan)
rDNA_mRNA_e2_shan_s <- cor(exp2_rDNA_div$shan.value,mRNA_exp2_shan, method = "spearman")
rDNA_mRNA_e2_shan_l <- sqrt(summary(lm(exp2_rDNA_div$shan.value~mRNA_exp2_shan))$adj.r.squared)

rRNA_mRNA_e2_shan_p <- cor(exp2_rRNA_div$shan.value,mRNA_exp2_shan)
rRNA_mRNA_e2_shan_s <- cor(exp2_rRNA_div$shan.value,mRNA_exp2_shan, method = "spearman")
rRNA_mRNA_e2_shan_l <- sqrt(summary(lm(exp2_rRNA_div$shan.value~mRNA_exp2_shan))$adj.r.squared)

model_rich <- rep(c(0.261844334, 0.318322406,0.384583649, 0.435570161,0.501653578, 0.579202516),each=3)

rDNA_model_e1_rich_p <- cor(exp1_rDNA_div$rich.value,model_rich)
rDNA_model_e1_rich_s <- cor(exp1_rDNA_div$rich.value,model_rich, method = "spearman")
rDNA_model_e1_rich_l <- sqrt(summary(lm(exp1_rDNA_div$rich.value~model_rich))$adj.r.squared)

rRNA_model_e1_rich_p <- cor(exp1_rRNA_div$rich.value,model_rich)
rRNA_model_e1_rich_s <- cor(exp1_rRNA_div$rich.value,model_rich, method = "spearman")
rRNA_model_e1_rich_l <- sqrt(summary(lm(exp1_rRNA_div$rich.value~model_rich))$adj.r.squared)

rDNA_model_e2_rich_p <- cor(exp2_rDNA_div$rich.value,model_rich)
rDNA_model_e2_rich_s <- cor(exp2_rDNA_div$rich.value,model_rich, method = "spearman")
rDNA_model_e2_rich_l <- sqrt(summary(lm(exp2_rDNA_div$rich.value~model_rich))$adj.r.squared)

rRNA_model_e2_rich_p <- cor(exp2_rRNA_div$rich.value,model_rich)
rRNA_model_e2_rich_s <- cor(exp2_rRNA_div$rich.value,model_rich, method = "spearman")
rRNA_model_e2_rich_l <- sqrt(summary(lm(exp2_rRNA_div$rich.value~model_rich))$adj.r.squared)

model_shan <- rep(c(2.56, 2.76, 2.90, 3.02, 3.11, 3.20),each=3)

rDNA_model_e1_shan_p <- cor(exp1_rDNA_div$shan.value,model_shan)
rDNA_model_e1_shan_s <- cor(exp1_rDNA_div$shan.value,model_shan, method = "spearman")
rDNA_model_e1_shan_l <- sqrt(summary(lm(exp1_rDNA_div$shan.value~model_shan))$adj.r.squared)

rRNA_model_e1_shan_p <- cor(exp1_rRNA_div$shan.value,model_shan)
rRNA_model_e1_shan_s <- cor(exp1_rRNA_div$shan.value,model_shan, method = "spearman")
rRNA_model_e1_shan_l <- sqrt(summary(lm(exp1_rRNA_div$shan.value~model_shan))$adj.r.squared)

rDNA_model_e2_shan_p <- cor(exp2_rDNA_div$shan.value,model_shan)
rDNA_model_e2_shan_s <- cor(exp2_rDNA_div$shan.value,model_shan, method = "spearman")
rDNA_model_e2_shan_l <- sqrt(summary(lm(exp2_rDNA_div$shan.value~model_shan))$adj.r.squared)

rRNA_model_e2_shan_p <- cor(exp2_rRNA_div$shan.value,model_shan)
rRNA_model_e2_shan_s <- cor(exp2_rRNA_div$shan.value,model_shan, method = "spearman")
rRNA_model_e2_shan_l <- sqrt(summary(lm(exp2_rRNA_div$shan.value~model_shan))$adj.r.squared)

#==============================================
# Establish layout grid
#==============================================
lay <- rbind(c(1,1,1,1,2,2,2,2,7),
             c(24,24,25,25,26,26,27,27,7),
             c(24,24,25,25,26,26,27,27,7),
             c(24,24,25,25,26,26,27,27,7),
             c(24,24,25,25,26,26,27,27,7),
             c(24,24,25,25,26,26,27,27,5),
             c(24,24,25,25,26,26,27,27,6),
             c(24,24,25,25,26,26,27,27,8),
             c(24,24,25,25,26,26,27,27,8),
```

```
             c(24,24,25,25,26,26,27,27,8),
             c(24,24,25,25,26,26,27,27,8),
             c(24,24,25,25,26,26,27,27,8),
             c(28,28,29,30,30,30,31,32,23),
             c(32,32,32,32,32,32,32,32,23),
             c(3,3,3,3,4,4,4,4,23),
             c(3,3,3,3,4,4,4,4,23),
             c(3,3,3,3,4,4,4,4,23),
             c(3,3,3,3,4,4,4,4,23),
             c(3,3,3,3,4,4,4,4,10),
             c(3,3,3,3,4,4,4,4,11),
             c(3,3,3,3,4,4,4,4,12),
             c(3,3,3,3,4,4,4,4,13),
             c(3,3,3,3,4,4,4,4,14),
             c(3,3,3,3,4,4,4,4,19),
             c(3,3,3,3,4,4,4,4,17),
             c(3,3,3,3,4,4,4,4,18),
             c(3,3,3,3,4,4,4,4,16),
             c(3,3,3,3,4,4,4,4,20),
             c(3,3,3,3,4,4,4,4,21),
             c(3,3,3,3,4,4,4,4,22),
             c(3,3,3,3,4,4,4,4,39),
             c(3,3,3,3,4,4,4,4,15),
             c(3,3,3,3,4,4,4,4,40),
             c(3,3,3,3,4,4,4,4,9),
             c(3,3,3,3,4,4,4,4,33),
             c(3,3,3,3,4,4,4,4,33),
             c(3,3,3,3,4,4,4,4,33),
             c(34,34,35,36,36,36,37,38,33))

#===============================================
# Plot for primer B2
#===============================================
pdf("Figure 2.pdf", height = 7, width = 12)
grid.arrange(title_exp1, title_exp2, bar_e1_b2, bar_e2_b2, leg_cDNA,
             leg_gDNA, blank, blank, planc, bdel,
             burk, flav, myx, nitro, lacto,
             sphingo, rhodob, rhodo, psd, thio,
             ver, all_else, blank, Exp1_B2_richness_obs, Exp1_B2_richness_sha, Exp2_B2_richness_obs,
Exp2_B2_richness_sha, blank,
             xaxis_lab, blank, xaxis_lab, blank, blank, blank, xaxis_lab, blank, xaxis_lab, blank, cald,
micro,
             ncol=2, layout_matrix = lay)
dev.off()

#===============================================
# Plot for primer B1
#===============================================
pdf("Supplemental 16S Figure.pdf", height = 7, width = 12)
grid.arrange(title_exp1, title_exp2, bar_e1_b1, bar_e2_b1, leg_cDNA,
             leg_gDNA, blank, blank, planc, bdel,
             burk, flav, myx, nitro, lacto,
             sphingo, rhodob, rhodo, psd, thio,
             ver, all_else, blank, Exp1_B2_richness_obs, Exp1_B2_richness_sha, Exp2_B2_richness_obs,
Exp2_B2_richness_sha, blank,
             xaxis_lab, blank, xaxis_lab, blank, blank, blank, xaxis_lab, blank, xaxis_lab, blank, cald,
micro,
             ncol=2, layout_matrix = lay)
dev.off()

#==========================================================================================================
===
```

Supplemental File 2 : In silico quality filtering, rRNA filtering, and annotation of the mRNA

All commands detailed here are able to be run on a Linux operating system with an Ubuntu distribution.

The first step in the analysis pipeline is to check the quality of the raw datafiles using the FastQC program:

```
fastqc *fastq.gz
```

Thereafter, trimmomatic is used to clip the adapter sequences within reads and remove sequences of low quality:

```
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67632_H5WH2BGX3_1_exp1R1_rna_epidemiology__ATCACG_S1_R1_001_MM_1.fastq.gz
Exp1_R1_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R1_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67633_H5WH2BGX3_1_exp1R2_rna_epidemiology__CGATGT_S2_R1_001_MM_1.fastq.gz
Exp1_R2_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R2_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67634_H5WH2BGX3_1_exp1R3_rna_epidemiology__TTAGGC_S3_R1_001_MM_1.fastq.gz
Exp1_R3_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R3_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67635_H5WH2BGX3_1_exp1R4_rna_epidemiology__TGACCA_S4_R1_001_MM_1.fastq.gz
Exp1_R4_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R4_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67636_H5WH2BGX3_1_exp1R5_rna_epidemiology__ACAGTG_S5_R1_001_MM_1.fastq.gz
Exp1_R5_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R5_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67637_H5WH2BGX3_1_exp1R6_rna_epidemiology__GCCAAT_S6_R1_001_MM_1.fastq.gz
Exp1_R6_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp1_R6_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67638_H5WH2BGX3_1_exp2R1_rna_epidemiology__CAGATC_S7_R1_001_MM_1.fastq.gz
Exp2_R1_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp2_R1_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67639_H5WH2BGX3_1_exp2R2_rna_epidemiology__ACTTGA_S8_R1_001_MM_1.fastq.gz
```

```
Exp2_R2_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp2_R2_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67640_H5WH2BGX3_1_exp2R3_rna_epidemiology__GATCAG_S9_R1_001_MM_1.fastq.gz
Exp2_R3_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp2_R3_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67641_H5WH2BGX3_1_exp2R4_rna_epidemiology__TAGCTT_S10_R1_001_MM_1.fastq.gz
Exp2_R4_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp2_R4_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67642_H5WH2BGX3_1_exp2R5_rna_epidemiology__GGCTAC_S11_R1_001_MM_1.fastq.gz
Exp2_R5_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp2_R5_trim_log.txt &
nohup java -jar /home/utoxadmin/Desktop/BioStatsPrograms/Trimmomatic-0.33/trimmomatic-0.33.jar
SE -threads 4
BSSE_QGF_67643_H5WH2BGX3_1_exp2R6_rna_epidemiology__CTTGTA_S12_R1_001_MM_1.fastq.gz
Exp2_R6_trimmed.fq.gz ILLUMINACLIP:TruSeq3-SE.fa:6:30:5 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36 >& Exp2_R6_trim_log.txt &
```

The quality of the trimmed datafiles are again checked using the FastQC program:

```
fastqc *trimmed.fq.gz
```

After ensuring that the rRNA databases are indexed, SortMeRNA is run to remove remaining rRNA encoding reads (an example is provided below for Experiment 1 Reactor 1):

```
gzip -d -k Exp1_R1_trimmed.fq.gz
time ./sortmerna --ref ./rRNA_databases/silva-bac-16s-id90.fasta,./index/silva-bac-
16s-db:./rRNA_databases/silva-bac-23s-id98.fasta,./index/silva-bac-23s-
db:./rRNA_databases/silva-arc-16s-id95.fasta,./index/silva-arc-16s-
db:./rRNA_databases/silva-arc-23s-id98.fasta,./index/silva-arc-23s-
db:./rRNA_databases/silva-euk-18s-id95.fasta,./index/silva-euk-18s-
db:./rRNA_databases/silva-euk-28s-id98.fasta,./index/silva-euk-
28s:./rRNA_databases/rfam-5s-database-id98.fasta,./index/rfam-5s-
db:./rRNA_databases/rfam-5.8s-database-id98.fasta,./index/rfam-5.8s-db --reads
Exp1_R1_trimmed.fq --sam --num_alignments 1 --fastx --aligned Exp1_R1rRNA --other
E1_R1_s_mRNA --log -v -a 20
rm Exp1_R1_trimmed.fq
rm Exp1_R1rRNA.fq
rm Exp1_R1rRNA.sam
```

The files are unzipped and converted from fastq to fasta format:

```
nohup gzip -k E1_R1_s_mRNA.fq >& log1.out &
nohup gzip -k E1_R2_s_mRNA.fq >& log2.out &
nohup gzip -k E1_R3_s_mRNA.fq >& log3.out &
nohup gzip -k E1_R4_s_mRNA.fq >& log4.out &
nohup gzip -k E1_R5_s_mRNA.fq >& log5.out &
nohup gzip -k E1_R6_s_mRNA.fq >& log6.out &

nohup gzip -k E2_R1_s_mRNA.fq >& log7.out &
```

```
nohup gzip -k E2_R2_s_mRNA.fq >& log8.out &
nohup gzip -k E2_R3_s_mRNA.fq >& log9.out &
nohup gzip -k E2_R4_s_mRNA.fq >& log10.out &
nohup gzip -k E2_R5_s_mRNA.fq >& log11.out &
nohup gzip -k E2_R6_s_mRNA.fq >& log12.out &

cat E1_R1_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E1_R1_s_mRNA.fasta
cat E1_R2_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E1_R2_s_mRNA.fasta
cat E1_R3_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E1_R3_s_mRNA.fasta
cat E1_R4_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E1_R4_s_mRNA.fasta
cat E1_R5_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E1_R5_s_mRNA.fasta
cat E1_R6_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E1_R6_s_mRNA.fasta

cat E2_R1_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E2_R1_s_mRNA.fasta
cat E2_R2_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E2_R2_s_mRNA.fasta
cat E2_R3_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E2_R3_s_mRNA.fasta
cat E2_R4_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E2_R4_s_mRNA.fasta
cat E2_R5_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E2_R5_s_mRNA.fasta
cat E2_R6_s_mRNA.fq | perl -e
'$i=0;while(<>){if(/^@/&&$i==0){s/^@/>/;print;}elsif($i==1){print;$i=-3}$i++;}' >
E2_R6_s_mRNA.fasta

rm *s_mRNA.fq
```

The appropriate database was downloaded from www.uniprot.org on March 6th, 2018 by either (1) searching for ec:* or (2) downloading the uniprot-all database. Both the TreMBL and SwissProt sequences were included in the database. The format of the downloaded files was given in three columns: protein ID, EC#, and fasta sequence. After the download was complete, the files were unzipped. The database was processed as follows:

```
sed -i -e 's/^/>/' NEW_EC_Library.tab
sed -i -e 's/\t/\n/2' NEW_EC_Library.tab
sed -i -e 's/ /_/g' NEW_EC_Library.tab
sed -i -e 's/\t/\?/g' NEW_EC_Library.tab
wc -l NEW_EC_Library.tab
tail -n+3 NEW_EC_Library.tab > NEW_EC_Library.fasta
```

The DIAMOND software was used to run the annotation. The first step of this annotation was to compile the database:

```
diamond makedb --in NEW_EC_Library.fasta -d EC_NEW
```

The reactors were annotated with a minimum bitscore cutoff of 50 into a tabular output format:

```
diamond blastx --db EC_NEW --threads 19 --out
exp1R1_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R1_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp1R2_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R2_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp1R3_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R3_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp1R4_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R4_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp1R5_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R5_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp1R6_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E1_R6_s_mRNA.fasta --max-
target-seqs 1 --min-score 50

diamond blastx --db EC_NEW --threads 19 --out
exp2R1_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R1_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp2R2_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R2_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp2R3_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R3_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp2R4_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R4_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp2R5_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R5_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
diamond blastx --db EC_NEW --threads 19 --out
exp2R6_EC_uniprot_diamond_fullEC_NEW.out --outfmt 6 -q E2_R6_s_mRNA.fasta --max-
target-seqs 1 --min-score 50
```

The files were further processed for downstream analyses:

```
sed -i 's/\?/\?_/g' *diamond_fullEC_NEW.out
sed -i 's/\t/,/g' *diamond_fullEC_NEW.out
sed -i 's/;/,/g' *diamond_fullEC_NEW.out
```

For the collation of the taxonomic origin of the individual read annotations, the first step was to collect the counts per unique Uniprot identification. This step was accomplished by processing within the Linux Ubuntu shell:

```
sed -i 's/,_/\$_/g' *diamond_fullEC_NEW.out

cat exp1R1_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp1R1_counts_per_uniprotMarch2018.out
cat exp1R2_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp1R2_counts_per_uniprotMarch2018.out
cat exp1R3_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp1R3_counts_per_uniprotMarch2018.out
cat exp1R4_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp1R4_counts_per_uniprotMarch2018.out
cat exp1R5_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp1R5_counts_per_uniprotMarch2018.out
cat exp1R6_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp1R6_counts_per_uniprotMarch2018.out

cat exp2R1_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp2R1_counts_per_uniprotMarch2018.out
cat exp2R2_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp2R2_counts_per_uniprotMarch2018.out
cat exp2R3_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp2R3_counts_per_uniprotMarch2018.out
cat exp2R4_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp2R4_counts_per_uniprotMarch2018.out
cat exp2R5_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp2R5_counts_per_uniprotMarch2018.out
cat exp2R6_EC_uniprot_diamond_fullEC_NEW.out | cut -d ',' -f2 | sort | uniq -c | sed
-e 's/^[ \t]*//' > exp2R6_counts_per_uniprotMarch2018.out
```

The files were aggregated using the CondensedAllExperimentCountsperUniprot.R in R 3.5.1:

```
library("rtk"); packageVersion("rtk")      # At time of implementation, 0.2.5.3
library("data.table");packageVersion("data.table") # At time of implementation,
1.10.4
library('dplyr');   packageVersion("dplyr") # At time of implementation, 0.7.2
library('tidyr');   packageVersion("tidyr") # At time of implementation, 0.6.3

#Read in the EC Table, the first is the count value "name of reactor" and the second
is the uniprot-ECnumber combined "EC"
exp1R1 <-
as.data.table(read.table("exp1R1_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp1R1,c("exp1R1","EC"))
```

```
exp1R2 <-
as.data.table(read.table("exp1R2_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp1R2,c("exp1R2","EC"))
exp1R3 <-
as.data.table(read.table("exp1R3_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp1R3,c("exp1R3","EC"))
exp1R4 <-
as.data.table(read.table("exp1R4_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp1R4,c("exp1R4","EC"))
exp1R5 <-
as.data.table(read.table("exp1R5_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp1R5,c("exp1R5","EC"))
exp1R6 <-
as.data.table(read.table("exp1R6_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp1R6,c("exp1R6","EC"))

exp2R1 <-
as.data.table(read.table("exp2R1_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp2R1,c("exp2R1","EC"))
exp2R2 <-
as.data.table(read.table("exp2R2_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp2R2,c("exp2R2","EC"))
exp2R3 <-
as.data.table(read.table("exp2R3_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp2R3,c("exp2R3","EC"))
exp2R4 <-
as.data.table(read.table("exp2R4_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp2R4,c("exp2R4","EC"))
exp2R5 <-
as.data.table(read.table("exp2R5_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp2R5,c("exp2R5","EC"))
exp2R6 <-
as.data.table(read.table("exp2R6_counts_per_uniprotMarch2018.out",header=FALSE, sep =
" ")); setnames(exp2R6,c("exp2R6","EC"))

#Set the uniprot-ECnumber combined "EC" as the key
setkey(exp1R1,"EC")
setkey(exp1R2,"EC")
setkey(exp1R3,"EC")
setkey(exp1R4,"EC")
setkey(exp1R5,"EC")
setkey(exp1R6,"EC")

setkey(exp2R1,"EC")
setkey(exp2R2,"EC")
setkey(exp2R3,"EC")
setkey(exp2R4,"EC")
setkey(exp2R5,"EC")
setkey(exp2R6,"EC")

#Merge the experiments
merged_exp1 <- merge(merge(merge(merge(merge(exp1R1, exp1R2, all=TRUE), exp1R3,
all=TRUE), exp1R4, all=TRUE), exp1R5, all = TRUE), exp1R6, all = TRUE)
merged_exp1_df <- as.data.frame(merged_exp1[,2:7])
```

```
merged_exp2 <- merge(merge(merge(merge(merge(exp2R1, exp2R2, all=TRUE), exp2R3,
all=TRUE), exp2R4, all=TRUE), exp2R5, all = TRUE), exp2R6, all = TRUE)
merged_exp2_df <- as.data.frame(merged_exp2[,2:7])

merged_all <- merge(merged_exp1, merged_exp2, all=TRUE)
merged_all_df <- as.data.frame(merged_all[,1:13])
merged_all_df[is.na(merged_all_df)] <- 0

#Split the UniprotID and EC
output <- merged_all_df %>%
  separate(EC, c("UniprotID", "EC"), "\\?_")

write.csv(output,"CondensedTableUniprot.txt", sep = "\t")
```

The appropriate database was downloaded from www.uniprot.org on March 6th, 2018 by searching for ec:*. Both the TreMBL and SwissProt sequences were included in the database. The format of the downloaded files was given in multiple columns: uniprot ID and the subsequent levels of taxonomic annotation (super kingdom, kingdom, phylum, class, order, family, genus). After the download was complete, the files were unzipped. The downloaded file was processed to ensure that the first separator was a tab and the remaining separators were semi-colons:

```
awk -F $'\t' '{print $1,"\t",$4,";",$5,";",$6,";",$7,";",$8,";",$9,";",$10}'
NEW_EC_Library.tab | sed 's/ \t/\t/g' > NEW_EC_uniprot.tab
```

The tab file was then converted into an SQLITE database for faster and more efficient processing downstream using the following TaxonomySQLiteCreationGenus.py python script:

```
import sqlite3
import csv
import sys

sqlite_file = 'Uniprot_Taxonomy_to_Genus.db'    # name of the sqlite database file
table_name2 = 'Uniprot_Taxonomy_to_Genus'  # name of the table to be created
id_field = 'id' # name of the column
taxon_field = 'taxon' # name of the column
field_type = 'TEXT'  # column data type

# Connecting to the database file
conn = sqlite3.connect(sqlite_file)
c = conn.cursor()

# Creating a second table with 1 column and set it as PRIMARY KEY
# note that PRIMARY KEY column must consist of unique values!
c.execute('CREATE TABLE {tn} ({nf} {ft} PRIMARY KEY, {cf} {ft})'\
        .format(tn=table_name2, nf=id_field, ft=field_type, cf=taxon_field))

filename = sys.argv[-1]
```

```
with open(filename, 'r') as f:
    reader = csv.reader(f, delimiter='\t')
    for row in reader:
        c.execute('INSERT INTO Uniprot_Taxonomy_to_Genus VALUES (?,?)', row)

# Committing changes and closing the connection to the database file
conn.commit()
conn.close()
```

Each EC number is processed individually from the condensed table:

```
grep -Pe 1\\.1\\.1\\.1[\$\"] CondensedTableUniprot.txt > 1.1.1.1_to_taxon_trans.txt
grep -Pe 1\\.1\\.1\\.10[\$\"] CondensedTableUniprot.txt > 1.1.1.10_to_taxon_trans.txt
grep -Pe 1\\.1\\.1\\.100[\$\"] CondensedTableUniprot.txt >
1.1.1.100_to_taxon_trans.txt
grep -Pe 1\\.1\\.1\\.101[\$\"] CondensedTableUniprot.txt >
1.1.1.101_to_taxon_trans.txt
(etc.)

python Taxonomy_Lookup.py 1.1.1.1_to_taxon_trans.txt > 1.1.1.1_taxon.txt
python Taxonomy_Lookup.py 1.1.1.10_to_taxon_trans.txt > 1.1.1.10_taxon.txt
python Taxonomy_Lookup.py 1.1.1.100_to_taxon_trans.txt > 1.1.1.100_taxon.txt
python Taxonomy_Lookup.py 1.1.1.101_to_taxon_trans.txt > 1.1.1.101_taxon.txt
(etc.)
```

Where the Taxonomy_lookup.py script is as follows:

```
import sqlite3
import csv
import sys

sqlite_file = 'Uniprot_Taxonomy_to_Genus.db'    # name of the sqlite database file
table_name2 = 'Uniprot_Taxonomy_output'  # name of the table to be created
id_field = 'id' # name of the column
taxon_field = 'taxon' # name of the column
field_type = 'TEXT'  # column data type

# Connecting to the database file
conn = sqlite3.connect(sqlite_file)
c = conn.cursor()

# Creating a second table with 1 column and set it as PRIMARY KEY
# note that PRIMARY KEY column must consist of unique values!

filename = sys.argv[-1]

with open(filename, 'r') as f:
    reader = csv.reader(f, delimiter=',')
    for row in reader:

        c.execute('SELECT * FROM Uniprot_Taxonomy_to_Genus WHERE id=?',(row[1],))
        xs = c.fetchall()
        for x in xs:
            print x[1]  + '\t' + row[3] + '\t' + row[4] + '\t' + row[5] + '\t' + row[6] + '\t' + row[7] +
'\t' + row[8] + '\t' + row[9] + '\t' + row[10] + '\t' + row[11] + '\t' + row[12]  + '\t' + row[13]  + '\t'
+ row[14]

# Committing changes and closing the connection to the database file
conn.commit()
conn.close()
```

The results were collated using the CondenseTaxonTable_fourths.R R-Script:

```
#===============================================================================
# Load the required libraries and print the version numbers
#===============================================================================
library("plyr");         packageVersion("plyr")          # At time of implementation, 1.8.4
library("ggplot2");      packageVersion("ggplot2")       # At time of implementation, 2.2.1
library("RColorBrewer"); packageVersion("RColorBrewer")  # At time of implementation, 1.1.2
library("rtk");          packageVersion("rtk")           # At time of implementation, 0.2.5.3
library("data.table");   packageVersion("data.table")    # At time of implementation, 1.10.4
library('vegan');        packageVersion("vegan")         # At time of implementation, 2.4.3
library('dplyr');        packageVersion("dplyr")         # At time of implementation, 0.7.2
library('tidyr');        packageVersion("tidyr")         # At time of implementation, 0.6.3

#===============================================================================
# Read in the list of EC Names
#===============================================================================
third_names <- read.table("Fourth_Taxon_Table_Names.txt", sep="\t", quote = "", row.names = NULL,
stringsAsFactors = FALSE, comment.char = "")

#===============================================================================
# Setup Summary Matrix that will hold the results and be printed at the end
#===============================================================================
summary <- matrix(ncol = length(third_names[,1]), nrow = 48)

#===============================================================================
# Loopover all EC Numbers
#===============================================================================
for (i in 1:length(third_names[,1])){
  print(paste("Analyzing ",third_names[i,1]))
  if (file.info(third_names[i,1])$size > 0){
  #===============================================================================
# Read in the EC Group
#===============================================================================
  tab <- read.table(third_names[i,1], sep="\t", quote = "", row.names = NULL, stringsAsFactors = FALSE,
comment.char = "")
  grepl('^;',tab[,1])
  #===============================================================================
  # Condense into unique taxons (sum) and write the condensed file
#===============================================================================
  tab2 <- ddply(tab, .(V1), numcolwise(sum))
  write.table(tab2, paste("condensed",third_names[i,1]), sep='$')
    #===============================================================================
  # Parse the tab2 dataframe into the exp1 and exp2 components for housekeeping
#===============================================================================
  tab2exp1_df <- tab2[,2:7]
  tab2exp2_df <- tab2[,8:13]
    #===============================================================================
  # Establish the minimum threshold to be considered detected
#===============================================================================
  threshhold <-
min(c(max(sum(tab2exp1_df[,1]),1),max(sum(tab2exp1_df[,2]),1),max(sum(tab2exp1_df[,3]),1),max(sum(tab2exp1
_df[,4]),1),max(sum(tab2exp1_df[,5]),1),max(sum(tab2exp1_df[,6]),1),

max(sum(tab2exp2_df[,1]),1),max(sum(tab2exp2_df[,2]),1),max(sum(tab2exp2_df[,3]),1),max(sum(tab2exp2_df[,4
]),1),max(sum(tab2exp2_df[,5]),1),max(sum(tab2exp2_df[,6]),1)
                      ))
   #===============================================================================
  # Calculate the diversity metrics (richness, Shannon, total number) for each of the looped over third
orders
  # process experiment 1  #===========================================================================
  summary[1,i] <- sum(tab2exp1_df[,1]/sum(tab2exp1_df[,1]) >= 1/threshhold)
  summary[2,i] <- sum(tab2exp1_df[,2]/sum(tab2exp1_df[,2]) >= 1/threshhold)
  summary[3,i] <- sum(tab2exp1_df[,3]/sum(tab2exp1_df[,3]) >= 1/threshhold)
  summary[4,i] <- sum(tab2exp1_df[,4]/sum(tab2exp1_df[,4]) >= 1/threshhold)
  summary[5,i] <- sum(tab2exp1_df[,5]/sum(tab2exp1_df[,5]) >= 1/threshhold)
  summary[6,i] <- sum(tab2exp1_df[,6]/sum(tab2exp1_df[,6]) >= 1/threshhold)
  summary[13,i] <- -sum(tab2exp1_df[,1]/sum(tab2exp1_df[,1])*log2(tab2exp1_df[,1]/sum(tab2exp1_df[,1])),
na.rm = TRUE)
  summary[14,i] <- -sum(tab2exp1_df[,2]/sum(tab2exp1_df[,2])*log2(tab2exp1_df[,2]/sum(tab2exp1_df[,2])),
na.rm = TRUE)
```

```
  summary[15,i] <- -sum(tab2exp1_df[,3]/sum(tab2exp1_df[,3])*log2(tab2exp1_df[,3]/sum(tab2exp1_df[,3])),
na.rm = TRUE)
  summary[16,i] <- -sum(tab2exp1_df[,4]/sum(tab2exp1_df[,4])*log2(tab2exp1_df[,4]/sum(tab2exp1_df[,4])),
na.rm = TRUE)
  summary[17,i] <- -sum(tab2exp1_df[,5]/sum(tab2exp1_df[,5])*log2(tab2exp1_df[,5]/sum(tab2exp1_df[,5])),
na.rm = TRUE)
  summary[18,i] <- -sum(tab2exp1_df[,6]/sum(tab2exp1_df[,6])*log2(tab2exp1_df[,6]/sum(tab2exp1_df[,6])),
na.rm = TRUE)
  summary[25,i] <- sum(tab2exp1_df[,1])
  summary[26,i] <- sum(tab2exp1_df[,2])
  summary[27,i] <- sum(tab2exp1_df[,3])
  summary[28,i] <- sum(tab2exp1_df[,4])
  summary[29,i] <- sum(tab2exp1_df[,5])
  summary[30,i] <- sum(tab2exp1_df[,6])
  #============================================================================
  # Search for those reads originating from bacteria
#============================================================================
  summary[37,i] <- sum(tab2exp1_df[,1]*grepl('Bacteria ;',tab2[,1]))
  summary[38,i] <- sum(tab2exp1_df[,2]*grepl('Bacteria ;',tab2[,1]))
  summary[39,i] <- sum(tab2exp1_df[,3]*grepl('Bacteria ;',tab2[,1]))
  summary[40,i] <- sum(tab2exp1_df[,4]*grepl('Bacteria ;',tab2[,1]))
  summary[41,i] <- sum(tab2exp1_df[,5]*grepl('Bacteria ;',tab2[,1]))
  summary[42,i] <- sum(tab2exp1_df[,6]*grepl('Bacteria ;',tab2[,1]))
  #============================================================================
  # process experiment 2  #=============================================================================
  summary[7,i]  <- sum(tab2exp2_df[,1]/sum(tab2exp2_df[,1]) >= 1/threshhold)
  summary[8,i]  <- sum(tab2exp2_df[,2]/sum(tab2exp2_df[,2]) >= 1/threshhold)
  summary[9,i]  <- sum(tab2exp2_df[,3]/sum(tab2exp2_df[,3]) >= 1/threshhold)
  summary[10,i] <- sum(tab2exp2_df[,4]/sum(tab2exp2_df[,4]) >= 1/threshhold)
  summary[11,i] <- sum(tab2exp2_df[,5]/sum(tab2exp2_df[,5]) >= 1/threshhold)
  summary[12,i] <- sum(tab2exp2_df[,6]/sum(tab2exp2_df[,6]) >= 1/threshhold)
  summary[19,i] <- -sum(tab2exp2_df[,1]/sum(tab2exp2_df[,1])*log2(tab2exp2_df[,1]/sum(tab2exp2_df[,1])),
na.rm = TRUE)
  summary[20,i] <- -sum(tab2exp2_df[,2]/sum(tab2exp2_df[,2])*log2(tab2exp2_df[,2]/sum(tab2exp2_df[,2])),
na.rm = TRUE)
  summary[21,i] <- -sum(tab2exp2_df[,3]/sum(tab2exp2_df[,3])*log2(tab2exp2_df[,3]/sum(tab2exp2_df[,3])),
na.rm = TRUE)
  summary[22,i] <- -sum(tab2exp2_df[,4]/sum(tab2exp2_df[,4])*log2(tab2exp2_df[,4]/sum(tab2exp2_df[,4])),
na.rm = TRUE)
  summary[23,i] <- -sum(tab2exp2_df[,5]/sum(tab2exp2_df[,5])*log2(tab2exp2_df[,5]/sum(tab2exp2_df[,5])),
na.rm = TRUE)
  summary[24,i] <- -sum(tab2exp2_df[,6]/sum(tab2exp2_df[,6])*log2(tab2exp2_df[,6]/sum(tab2exp2_df[,6])),
na.rm = TRUE)
  summary[31,i] <- sum(tab2exp2_df[,1])
  summary[32,i] <- sum(tab2exp2_df[,2])
  summary[33,i] <- sum(tab2exp2_df[,3])
  summary[34,i] <- sum(tab2exp2_df[,4])
  summary[35,i] <- sum(tab2exp2_df[,5])
  summary[36,i] <- sum(tab2exp2_df[,6])
  summary[43,i] <- sum(tab2exp2_df[,1]*grepl('Bacteria ;',tab2[,1]))
  summary[44,i] <- sum(tab2exp2_df[,2]*grepl('Bacteria ;',tab2[,1]))
  summary[45,i] <- sum(tab2exp2_df[,3]*grepl('Bacteria ;',tab2[,1]))
  summary[46,i] <- sum(tab2exp2_df[,4]*grepl('Bacteria ;',tab2[,1]))
  summary[47,i] <- sum(tab2exp2_df[,5]*grepl('Bacteria ;',tab2[,1]))
  summary[48,i] <- sum(tab2exp2_df[,6]*grepl('Bacteria ;',tab2[,1]))
  } else {summary[1:48,i] <- 0}
  }
#============================================================================
# Write the full final table
#============================================================================
write.table(t(summary), "Summary of all Fourths with Count Rerun March 12 2018 at least one read with
bacteria fraction.txt")
```

Supplemental File 3 : Sequencing Batch Reactor Monod model for multiple-species

```
#  The following is an R script that can be run on R v 3.0 or higher. Only the package deSolve
is required.

library(deSolve)
#===============================================================================================
# Set the ranges of the ecological umax and Ks values
#===============================================================================================
u_eco_max <- 9.8
u_eco_min <- 0.2

#===============================================================================================
# Set the bulk Ks value
#===============================================================================================
Ks_bulk <- 50

#==============================================================================================
# Select MRTs to use in the differential equations, establish certain required variables.
# in this case, MRT of 1 day is used as the example.
#==============================================================================================
RW1 <- c(); j <- 1; diversity <- c()

u_bulk = 5

# Use only one set of the u_s below for the solution carried forward, comment out all others
with #

# Growth Rates for MRT 1 day
u9 = 4.626275;u8 = 4.719709;u7 = 4.813127;u6 = 4.906577;u5 = u_bulk;u4 = 5.093445;u3 =
5.186879;u2 = 5.280314;u1 = 5.373748

# Growth Rates for MRT 3 day
# u9 = 4.054179;u8 = 4.290640;u7 = 4.527102;u6 = 4.763563;u5 = u_bulk;u4 = 5.236486;u3 =
5.472947;u2 = 5.709408;u1 = 5.945870
# Growth Rates for MRT 5 day
# u9 = 3.597233;u8 = 3.947921;u7 = 4.298609;u6 = 4.649298;u5 = u_bulk;u4 = 5.350675;u3 =
5.701365;u2 = 6.052054;u1 = 6.402743
# Growth Rates for MRT 7 day
# u9 = 3.288019;u8 = 3.716036;u7 = 4.144053;u6 = 4.572071;u5 = u_bulk;u4 = 5.428105;u3 =
5.856122;u2 = 6.284140;u1 = 6.712157
# Growth Rates for MRT 10 day
# u9 = 2.912718;u8 = 3.434623;u7 = 3.956528;u6 = 4.478434;u5 = u_bulk;u4 = 5.522244;u3 =
6.044149;u2 = 6.566055;u1 = 7.087960
# Growth Rates for MRT 15 day
# u9 = 2.492964;u8 = 3.119928;u7 = 3.746893;u6 = 4.373858;u5 = u_bulk;u4 = 5.627788;u3 =
6.254753;u2 = 6.881717;u1 = 7.508682

Ks9 = Ks_bulk;Ks8 = Ks_bulk;Ks7 = Ks_bulk;Ks6 = Ks_bulk;Ks5 = Ks_bulk;Ks4 = Ks_bulk;Ks3 =
Ks_bulk;Ks2 = Ks_bulk;Ks1 = Ks_bulk
b1=0.2; b2=0.1775 ;b3=0.155; b4=0.1325; b5=0.11; b6=0.0875; b7=0.065; b8=0.0425; b9=0.02

#===============================================================================================
# Set the parameters of the SBR
#===============================================================================================
Volume  = 12                # Volume of reactor
days_to_model = 36          # Time to run the integration over
sbr_seq = 1/6               # Given in days
V_withdrawn = 4             # in Liters
S1in    = 250               # Substrate inflow rate
Ys1S1   = 0.3               # Yield of bacterial growth (fixed)
inflow_time = 20/(60*24)    # Sequencing Batch Reactor Time step for the inflow
```

```
outflow_time = 10/(60*24)   # Sequencing Batch Reactor Time step for the outflow (drain)
inflow_outflow_time = inflow_time + outflow_time
settle_time = 30/(60*24)    # Sequencing Batch Reactor Time step for the settling
sludge_discharge_time = 10/(60*24)  # Sequencing Batch Reactor Time step for the sludge removal
integration_time_step = 0.5/(60*24) # Integration time step (set at useful interval)
Inflow = 0
Outflow = 0

#---------------------------------------------------------------------------------------------
# In acknowledgement that the assumption that changing one unit in range normalized u_max
# versus b_e values is equivalent is unlikely, a scaling parameter is established. When the
value is
# <1, then the u_max contributes more to the combination of growth parameters. When the value
is >1,
# the b_e contributes more. In this case, the value is set to 0.25 to assume
# that the u_max value range contributes more than the b_e. Values of 0.1
# to 0.8 were explored, and the major influence was on the magnitude of the difference in
richness and
# diversity across the MRT gradient, not the shape of the underlying relationship. Therefore,
0.25 was
# selected for efficient computational time. As the value approaches 1 or exceeds one, changes
in richness
# and diversity is minimized because the b_e range is saturated.
#---------------------------------------------------------------------------------------------
Scaling_Factor = 0.25

#---------------------------------------------------------------------------------------------
# How long the reactors are operated for (community equilibrium time)
times   <- seq(0, days_to_model, by = integration_time_step)

#==============================================================================================
# Parse the differential equations into a function entitles "SBR_Diff_Iter". Iterate to find a
# parametrically acceptable solution for the growth parameters that lead to persistence.
#==============================================================================================
repeat{ SBR_Diff_Iter <- function(t, state, parameters){
  with(as.list(c(state, parameters)),{
    Qfill <- inflow_on_func(t)
    Qclearweater_drain <- outflow_on_func(t)
    Qmixed_drain <- waste_rate_on_func(t)
    dV <- Qfill - Qclearweater_drain - Qmixed_drain
    dX1 <- (-X1/V * Qmixed_drain + u1 * (So)/(Ks1 + So) * X1 - b1 * X1)
    dX2 <- (-X2/V * Qmixed_drain + u2 * (So)/(Ks2 + So) * X2 - b2 * X2)
    dX3 <- (-X3/V * Qmixed_drain + u3 * (So)/(Ks3 + So) * X3 - b3 * X3)
    dX4 <- (-X4/V * Qmixed_drain + u4 * (So)/(Ks4 + So) * X4 - b4 * X4)
    dX5 <- (-X5/V * Qmixed_drain + u5 * (So)/(Ks5 + So) * X5 - b5 * X5)
    dX6 <- (-X6/V * Qmixed_drain + u6 * (So)/(Ks6 + So) * X6 - b6 * X6)
    dX7 <- (-X7/V * Qmixed_drain + u7 * (So)/(Ks7 + So) * X7 - b7 * X7)
    dX8 <- (-X8/V * Qmixed_drain + u8 * (So)/(Ks8 + So) * X8 - b8 * X8)
    dX9 <- (-X9/V * Qmixed_drain + u9 * (So)/(Ks9 + So) * X9 - b9 * X9)

    dmass <- dX1 + dX2 + dX3 + dX4 + dX5 + dX6 + dX7 + dX8 + dX9

    dSo <- (Qfill*SS1in/V - Qclearweater_drain*So/V - So/V * Qmixed_drain - u1 * (So)/(Ks1 +
(So)) * (X1) / Yield -
              u2 * (So)/(Ks2 + (So)) * (X2) / Yield -
              u3 * (So)/(Ks3 + (So)) * (X3) / Yield -
              u4 * (So)/(Ks4 + (So)) * (X4) / Yield -
              u5 * (So)/(Ks5 + (So)) * (X5) / Yield -
              u6 * (So)/(Ks6 + (So)) * (X6) / Yield -
              u7 * (So)/(Ks7 + (So)) * (X7) / Yield -
              u8 * (So)/(Ks8 + (So)) * (X8) / Yield -
              u9 * (So)/(Ks9 + (So)) * (X9) / Yield)
```

```
    list(c(dV, dSo, dX1,dX2,dX3,dX4,dX5,dX6,dX7,dX8,dX9, dmass))
  })
}

parameters500 <- c( SS1in = S1in, SBR = sbr_seq,
                    u1 = u1, u2 = u2, u3 = u3,  u4 = u4, u5 = u5,
                    Ks1 = Ks1, Ks2 = Ks2, Ks3 = Ks3,  Ks4 = Ks4, Ks5 = Ks5,
                    u6 = u6, u7 = u7, u8 = u8,  u9 = u9,
                    Ks6 = Ks6, Ks7 = Ks7, Ks8 = Ks8,  Ks9 = Ks9,
                    Yield = Ys1S1,
                    b1=b1,b2=b2, b3=b3, b4=b4, b5=b5, b6=b6, b7=b7, b8=b8, b9=b9)

#==============================================================================================
# Set the initial State
#==============================================================================================
n = 9
num_spec <- as.numeric(seq(1,1, length.out = n))
state <- c(V= Volume, So = 250, X = num_spec*100/8, mass = 100)

#==============================================================================================
# Establish the inflow/outflow/waste triggers to describe the SBR system, turn on the
appropriate functional set that
# matches the selected MRT above.
#==============================================================================================
#---------------------------------------------------------------------------------------------
#For 1d MRT
#---------------------------------------------------------------------------------------------
inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_time)/integration_time_step + 1),
                                                            rep(4/inflow_time,
(inflow_time)/integration_time_step + 1)), days_to_model/sbr_seq),0)))

inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")

outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time)/integration_time_step + 1), rep(2/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
days_to_model/sbr_seq),0)))

outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")

waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),

rep(2/sludge_discharge_time, sludge_discharge_time/integration_time_step + 1),
                                                                rep(0, (inflow_outflow_time
+ settle_time)/integration_time_step + 1)), days_to_model/sbr_seq),0)))

waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")

#---------------------------------------------------------------------------------------------
#For 3d MRT
#---------------------------------------------------------------------------------------------
#inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_time)/integration_time_step + 1),rep(4/inflow_time, (inflow_time)/integration_time_step
+ 1)), days_to_model/sbr_seq),0)))
#inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")

#outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time)/integration_time_step + 1), rep(2/outflow_time,
```

```
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1),rep(0,480),rep(0,480)), days_to_model*2),0)))
#outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")

#waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
rep(4/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
(inflow_outflow_time + settle_time)/integration_time_step + 1)), days_to_model/sbr_seq),0)))
#waste_rate_outflow <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
rep(2/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
(inflow_outflow_time + settle_time)/integration_time_step + 1),rep(0,480),rep(0,480)),
days_to_model*2),0)))
#waste_rate_on <- waste_rate_on - waste_rate_outflow
#waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")

#-----------------------------------------------------------------------------------------
#For 5d MRT
#-----------------------------------------------------------------------------------------
#inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_time)/integration_time_step + 1),rep(4/inflow_time, (inflow_time)/integration_time_step
+ 1)), days_to_model/sbr_seq),0)))
#inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")

#outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time)/integration_time_step + 1), rep(4/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
days_to_model/sbr_seq),0)))
#outflow_because_of_waste <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq
- inflow_outflow_time)/integration_time_step + 1), rep(1.2/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1),rep(0,480),rep(0,480)), days_to_model*2),0)))
#outflow_on[,2] <- outflow_on[,2] - outflow_because_of_waste[,2]
#outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")

#waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
rep(1.2/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
(inflow_outflow_time + settle_time)/integration_time_step + 1),rep(0,480),rep(0,480)),
days_to_model*2),0)))
#waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")

#-----------------------------------------------------------------------------------------
#For 7d MRT
#-----------------------------------------------------------------------------------------
#inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_time)/integration_time_step + 1),rep(4/inflow_time, (inflow_time)/integration_time_step
+ 1)), days_to_model/sbr_seq),0)))
#inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")

#outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time)/integration_time_step + 1), rep(4/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
days_to_model/sbr_seq),0)))
#outflow_because_of_waste <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq
- inflow_outflow_time)/integration_time_step + 1), rep(1.7/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1),rep(0,480*5)), days_to_model),0)))
#outflow_on[,2] <- outflow_on[,2] - outflow_because_of_waste[,2]
#outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")
```

```
#waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
rep(1.7/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
(inflow_outflow_time + settle_time)/integration_time_step + 1), rep(0,480*5)),
days_to_model),0)))
#waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")

#-----------------------------------------------------------------------------------------
#For 10d MRT
#-----------------------------------------------------------------------------------------
#inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_time)/integration_time_step + 1),rep(4/inflow_time, (inflow_time)/integration_time_step
+ 1)), days_to_model/sbr_seq),0)))
#inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")

#outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time)/integration_time_step + 1), rep(4/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
days_to_model/sbr_seq),0)))
#outflow_because_of_waste <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq
- inflow_outflow_time)/integration_time_step + 1), rep(1.2/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1),rep(0,480*5)), days_to_model),0)))
#outflow_on[,2] <- outflow_on[,2] - outflow_because_of_waste[,2]
#outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")

#waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
rep(1.2/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
(inflow_outflow_time + settle_time)/integration_time_step + 1), rep(0,480*5)),
days_to_model),0)))
#waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")

#-----------------------------------------------------------------------------------------
#For 15d MRT
#-----------------------------------------------------------------------------------------
#inflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_time)/integration_time_step + 1),rep(4/inflow_time, (inflow_time)/integration_time_step
+ 1)), days_to_model/sbr_seq),0)))
#inflow_on_func <- approxfun(inflow_on, rule = 2, method = "constant")

#outflow_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time)/integration_time_step + 1), rep(4/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step + 1)),
days_to_model/sbr_seq),0)))
#outflow_because_of_waste <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq
- inflow_outflow_time)/integration_time_step + 1), rep(0.8/outflow_time,
(outflow_time)/integration_time_step + 1), rep(0, (inflow_time)/integration_time_step +
1),rep(0,480*5)), days_to_model),0)))
#outflow_on[,2] <- outflow_on[,2] - outflow_because_of_waste[,2]
#outflow_on_func <- approxfun(outflow_on, rule = 2, method = "constant")

#waste_rate_on <- as.data.frame(list(times = times, import = c(rep(c(rep(0, (sbr_seq -
inflow_outflow_time - settle_time - sludge_discharge_time)/integration_time_step + 1),
rep(0.8/sludge_discharge_time, (sludge_discharge_time)/integration_time_step + 1), rep(0,
(inflow_outflow_time + settle_time)/integration_time_step + 1), rep(0,480*5)),
days_to_model),0)))
#waste_rate_on_func <- approxfun(waste_rate_on, rule = 2, method = "constant")

#==========================================================================================
# Run the differential equation
#==========================================================================================
```

```
RW1 <- ode(y = state, times = times, func = SBR_Diff_Iter, parms = parameters500,  method =
"bdf")

output_error <- RW1[480*6*10,]/RW1[480*6*15,]
abs_output_error <- abs(output_error-output_error[13])
u9 = unname(u9+(output_error[12]-output_error[13])-0.5*(output_error[8]-output_error[13]),
force =FALSE)
u8 = unname(u8+(output_error[11]-output_error[13])-0.5*(output_error[8]-output_error[13]),
force =FALSE)
u7 = unname(u7+(output_error[10]-output_error[13])-0.5*(output_error[8]-output_error[13]),
force =FALSE)
u6 = unname(u6+(output_error[9]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
=FALSE)
u5 = u_bulk
u4 = unname(u4+(output_error[7]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
=FALSE)
u3 = unname(u3+(output_error[6]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
=FALSE)
u2 = unname(u2+(output_error[5]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
=FALSE)
u1 = unname(u1+(output_error[4]-output_error[13])-0.5*(output_error[8]-output_error[13]), force
=FALSE)

u_s <- c(u1,u2,u3,u4,u5,u6,u7,u8,u9)

print(max(abs(output_error[4:13]-1)))
print(u_s)
plot(RW1, mfrow = c(3,5))

#Check whether the solution is in an acceptable error range
if (! max(abs(output_error[4:13]-1)) > 0.0001){break}
}

#===============================================================================================
# Plot the results of the optimization
#===============================================================================================
plot(RW1, mfrow = c(3,5),xlim=c(14,15))

#===============================================================================================
# Establish the differential equations for a SBR AS system as a string to be parsed
#===============================================================================================
b_s <- c(0.2, 0.1775, 0.155, 0.1325, 0.11, 0.0875, 0.065, 0.0425, 0.02)
umax_s <- c(u1,u2,u3,u4,u5,u6,u7,u8,u9)

n = round(sqrt(((max(umax_s)-min(umax_s))/(u_eco_max-u_eco_min))^2+(0.18/0.18*
Scaling_Factor)^2)*50)
fit <- lm(umax_s~b_s)

b_range <- seq(min(b_s),max(b_s), length.out = n)
umax_range <- fit$coefficients[2] * b_range + fit$coefficients[1] #+ fit2$coefficients[4]  *
rv$Kss1S1^3

dX_carb_eq  <- paste("dX", 1:n, " <- (-X",  1:n, "/V * Qmixed_drain + u",  1:n,
                     " * (So)/(Ks + So) * X",  1:n, " - b",  1:n," * X",  1:n, ")", sep="")

dmass_eq <- paste("dmass <- ", paste("dX", 1:n, sep="",collapse=" + "), sep="")

dS_eq <- paste("dSo <- (Qfill*SS1in/V - Qclearweater_drain*So/V - So/V * Qmixed_drain - ",
               paste("u", 1:n, " * (So)/(Ks + So) * (X", 1:n, ") / Yield",   sep="",collapse="
- "), ")", sep="")
```

```
state_list <- paste("list(c(dV, dSo, ", paste("dX",1:n, sep="", collapse = ","), ", dmass))",
sep="", collapse = "")

write(c(dX_carb_eq,dmass_eq,dS_eq,state_list),"equation_output.txt")

#===========================================================================================
# Parse the differential equations into a function entitles "SBR_Diff"
#===========================================================================================
# SBR_Diff <- function(t, state, parameters){
#  with(as.list(c(state, parameters)),{
#    Qfill <- inflow_on_func(t)
#    Qclearweater_drain <- outflow_on_func(t)
#    Qmixed_drain <- waste_rate_on_func(t)
#    dV <- Qfill - Qmixed_drain - Qclearweater_drain
#    eval(parse(text=dX_carb_eq))
#    eval(parse(text=dmass_eq))
#    eval(parse(text=dS_eq))
#    eval(parse(text=state_list))
#  })
#}
#===========================================================================================
# For large systems of equations, copy the text from the generated "equation_output" file and
attach below.
# For larger n species, the system will take a substantial amount of time to run
#===========================================================================================
SBR_Diff <- function(t, state, parameters){
  with(as.list(c(state, parameters)),{
    Qfill <- inflow_on_func(t)
    Qclearweater_drain <- outflow_on_func(t)
    Qmixed_drain <- waste_rate_on_func(t)
    dV <- Qfill - Qclearweater_drain - Qmixed_drain
    dX1 <- (-X1/V * Qmixed_drain + u1 * (So)/(Ks + So) * X1 - b1 * X1)
    dX2 <- (-X2/V * Qmixed_drain + u2 * (So)/(Ks + So) * X2 - b2 * X2)
    dX3 <- (-X3/V * Qmixed_drain + u3 * (So)/(Ks + So) * X3 - b3 * X3)
    dX4 <- (-X4/V * Qmixed_drain + u4 * (So)/(Ks + So) * X4 - b4 * X4)
    dX5 <- (-X5/V * Qmixed_drain + u5 * (So)/(Ks + So) * X5 - b5 * X5)
    dX6 <- (-X6/V * Qmixed_drain + u6 * (So)/(Ks + So) * X6 - b6 * X6)
    dX7 <- (-X7/V * Qmixed_drain + u7 * (So)/(Ks + So) * X7 - b7 * X7)
    dX8 <- (-X8/V * Qmixed_drain + u8 * (So)/(Ks + So) * X8 - b8 * X8)
    dX9 <- (-X9/V * Qmixed_drain + u9 * (So)/(Ks + So) * X9 - b9 * X9)
    dX10 <- (-X10/V * Qmixed_drain + u10 * (So)/(Ks + So) * X10 - b10 * X10)
    dX11 <- (-X11/V * Qmixed_drain + u11 * (So)/(Ks + So) * X11 - b11 * X11)
    dX12 <- (-X12/V * Qmixed_drain + u12 * (So)/(Ks + So) * X12 - b12 * X12)
    dX13 <- (-X13/V * Qmixed_drain + u13 * (So)/(Ks + So) * X13 - b13 * X13)

    dmass <- dX1 + dX2 + dX3 + dX4 + dX5 + dX6 + dX7 + dX8 + dX9 + dX10 + dX11 + dX12 + dX13

    dSo <- (Qfill*SS1in/V - Qclearweater_drain*So/V - So/V * Qmixed_drain - u1 * (So)/(Ks + So)
*
              (X1) / Yield - u2 * (So)/(Ks + So) * (X2) / Yield - u3 * (So)/(Ks + So) * (X3) /
Yield - u4 * (So)/(Ks + So) *
              (X4) / Yield - u5 * (So)/(Ks + So) * (X5) / Yield - u6 * (So)/(Ks + So) * (X6) /
Yield - u7 * (So)/(Ks + So) *
              (X7) / Yield - u8 * (So)/(Ks + So) * (X8) / Yield - u9 * (So)/(Ks + So) * (X9) /
Yield - u10 * (So)/(Ks + So) *
              (X10) / Yield - u11 * (So)/(Ks + So) * (X11) / Yield - u12 * (So)/(Ks + So) *
(X12) / Yield - u13 * (So)/(Ks + So) *
              (X13) / Yield)
    list(c(dV, dSo, dX1,dX2,dX3,dX4,dX5,dX6,dX7,dX8,dX9,dX10,dX11,dX12,dX13, dmass))  })
}
```

```
parameters_final <- c(b = b_range, SS1in = S1in, tcycle = sbr_seq, V_withdrawn_per_cycle =
V_withdrawn,
                      u = umax_range, Ks = Ks_bulk, Yield = Ys1S1)

#===============================================================================================
# Set the initial State
#===============================================================================================
num_spec <- as.numeric(seq(1,1, length.out = n))
state <- c(V= Volume, So = 250, X = num_spec*100/n, mass = 100)

#===============================================================================================
# Run the differential equation
#===============================================================================================
RW_final <- ode(y = state, times = times, func = SBR_Diff, parms = parameters_final)

#===============================================================================================
# Calculate the Shannon diversity metric
#===============================================================================================
mass_i <- n + 4
final_X <- n + 3
diversity_time <- (days_to_model/integration_time_step-1)
diversity <- sum(-1*RW_final[diversity_time,4:final_X]/(RW_final[diversity_time,mass_i]) *
                   log(RW_final[diversity_time,4:final_X]/(RW_final[diversity_time,mass_i])))
TSS <- RW_final[diversity_time,mass_i]

#===============================================================================================
# Plot the results, NOTE - not volume normalized here
#===============================================================================================
plot(RW_final)
plot(RW_final, mfrow = c(3,5),xlim=c(29,30))
plot(RW_final, mfrow = c(3,5),xlim=c(29,30),ylim=c(10,13))

#===============================================================================================
# Solve the same system of equations for an initial spiked state of So = 500 mg/L to determine
# the theoretical instantaneous oxygen uptake rate (OUR).
#===============================================================================================
Spike_state <- c(V= Volume, So = 500, X = as.numeric(RW_final[diversity_time,4:final_X]), mass
= TSS)
RW_spike <- ode(y = Spike_state, times = times[1:480], func = SBR_Diff, parms =
parameters_final)
init_time <- RW_spike[1:20,1]
init_subs <- RW_spike[1:20,3]
fit_spike <- lm(init_subs~init_time)
fit_spike
fit_spike$coefficients[2]/TSS

sum(as.numeric(RW_final[diversity_time,4:final_X])*umax_range)/TSS
```

Supplemental Materials 1 : Details of RNA and DNA isolation, handling, and sequencing protocols

Supplemental Materials 2 : Equivalent model solution for a continuously-stirred tank reactor (CSTR) with R code

Supplemental Materials 1 : Details of RNA and DNA isolation, handling, and sequencing protocols

RNA/DNA Isolation

In total, 10-mL acidic phenol-saturated water (pH 4.3; Sigma-Aldrich) was added to the frozen cell pellets. Thereof, 1 mL was transferred to a 2-mL tube containing 0.1-mm zirconia/silica beads and combined with 250-µl pH 5.1 buffer (0.4-ml 0.5 M EDTA, molecular biology grade (Invitrogen), 0.33-ml 3-M sodium acetate (pH 5.1), molecular biology grade (Sigma-Aldrich), 19.3-ml DEPC-treated water) and 100-µl 10% SDS (Sigma-Aldrich). After incubation at 65ºC for 2 min, the samples were bead beat at max speed for 2 min (MM300 TissueLyser, Qiagen Retsch), incubated at 65ºC for 8 min, bead beat at max speed for 2 min, and centrifuged at 14,000 × g at 4ºC for 5 min. The top layer was preserved for further processing.

Initially, 125 µl of phenol-saturated water (pH 4.3; Sigma-Aldrich) and 125 µl of chloroform:isoamylalcohol (24:1; Sigma-Aldrich) was added to the sample, and the mixture was shaken for 3 min, let stand for 3 min, and then centrifuged at 14,000 × g at 4ºC for 3 min. The top aqueous layer was transferred, and 100 µl of phenol and 100 µl of chloroform:isoamylalcohol (24:1) was added. The mixture was shaken for 3 min, let stood for 3 min, and then centrifuged at 14,000 × g at 4ºC for 3 min. The top aqueous layer was transferred, and 200 µl of chloroform:isoamylalcohol was added. The mixture was shaken for 3 min, let stood for 3 min, and then centrifuged at 14,000 x g at 4ºC for 3 min. The top layer was transferred, and 0.5 volume of 7.5-M ammonium acetate and 2 volumes of 100% ethanol was added to precipitate the pellet. The sample was mixed, spun down, incubated at -20ºC for at least 2 hours, and centrifuged at 21,000 × g for 30 min at 4ºC. After decanting the supernatant, the sample was rinsed with 200 µl of 80% ethanol and centrifuged at 21,000 × g for 10 min at 4ºC. After decanting the supernatant, the sample was vacuum centrifuged for 2-3 min. The RNA pellet was re-suspended with DEPC-treated RNase-free water to a total of 25 µl. Total nucleic acid concentration was measured on a Nanodrop (Invitrogen) or Qubit (Invitrogen), and aliquots of the samples were frozen at -80ºC.

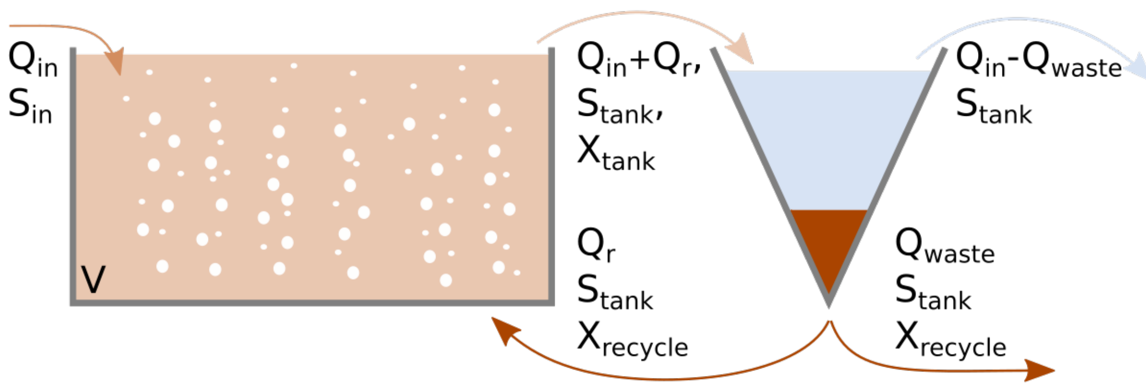
The protocol for extracting DNA followed the same procedure outlined above with the substitutions of pH 7.0 instead of pH 4.3 phenol-saturated water (Sigma Aldrich). The samples were quantified using Nanodrop (Invitrogen) or Qubit (Invitrogen), and the final samples were further cleaned using the DNA PowerCleanup PRO Kit (Qiagen) following the manufacturer's instructions. The purified DNA was frozen at -20ºC until further use.

The RNA samples were further cleaned using the TURBO DNase Kit. and the MoBIO RNA Pro Clean-Up Kit (MoBio) following the manufacturer's protocols. After reverse-transcription using the Superscript III kit (Invitrogen) the quality of the RNA was determined using a Bioanalzyer 2000 (Agilent Technologies) or TapeStation (Agilent Technologies), and the samples were stored at -80°C until further use.

16S Library Preparation and Sequencing

The initial polymerase chain reaction (PCR) consisted of (in triplication for each sample) 12.5 µl of 2X KAPA HiFi Hot Start Ready Mix, 0.75 µl forward primer (10 µM), 0.75 µl reverse primer (10 µM), 1.25 µl of DMSO, 9 µl of PCR grade water, and 1 µl of template DNA. Two sets of 16S rRNA primers (Integrated DNA Technologies, Inc., Skokie, Illinois, USA,) were used to amplify either the sample cDNA or gDNA to minimize the potential for probe bias [Guo et al., 2013]. The details of primers B1 and B2 are detailed in Supplemental Table 1. The PCR was run on a thermal cycler using the following program: 95ºC for 5 min; cycles of 98ºC for 20 seconds, 53ºC for 15 seconds, and 72ºC for 15 seconds (18 and 20 cycles for B1 and B2, respectively); 72ºC for 5 minutes. The samples were then cleaned using AMPure Beads following the manufacturer's instructions. Nextera XT index primers (N7XX and S5XX; Illumina) were attached to the amplicons in a subsequent PCR: 25 µl of 2X KAPA HiFi, 5 µl each of Nextera XT index primer 1 and 2, and 15 µl of the cleaned amplicon run at 95ºC for 3 min; 9 cycles of 95ºC for 30 seconds, 55ºC for 30 seconds, and 72ºC for 30 seconds; 72ºC for 5 minutes. The samples were again cleaned using AMPure Beads following the manufacturer's instructions. The samples were quantified and qualified using a Nanodrop (Invitrogen), Qubit (ThermoFisher), and Tapestation (Agilent). Samples were then normalized and pooled. Subsequently, the samples were sequenced using the PE 300 method on a MiSeq platform (Illumina) at the Genomics Diversity Center at the ETH in Zurich, Switzerland. The raw data is available at EMBL-EBI under the study number ERP024418.

Supplemental Materials 2 : Equivalent model solution for a continuously-stirred tank reactor (CSTR) with R code



SM2 Figure 1. Diagram of the continuously stirred tank reactor (CSTR). Q_{waste} and $Q_{recycle}$ are the waste and recycle flow rates, respectively, and all other parameters are defined in the main text. Q_{waste} is determined by the MRT.

To establish the continuously stirred tank reactor model the S^{*}_{min} Equation 1 in the main text can be rearranged to solve for either the u_{max} or b_e parameter:

$$\mu_{max} = \frac{\left(1+b_e*MRT*\left(1+\frac{S^*}{K_S}\right)\right)\frac{K_S}{S^*}+1}{MRT} \qquad \text{SM Eq. 1}$$

$$b_e = \frac{\frac{S^*}{K_S}(\mu_{max}*MRT-1)-1}{MRT(1+\frac{S^*}{K_S})} \qquad \text{SM Eq. 2}$$

where all parameters have been previously described in the main text. As the MRT increases, these parameters are checked that they satisfy the ecological maximum and minimum defined; when they violate a parameter, they are appropriately replaced.

Conventional differential equations are established to describe the CSTR with recycle:

$$\frac{dS}{dt} = \left(\frac{(S_{in}-S)*Q_{in}}{V} - \sum_{i=1}^{n}\left[\frac{\mu_{max,i}*S}{K_s+S} - b_{e,i}\right] * \frac{X_i}{Y}\right) \qquad \text{SM Eq. 3}$$

$$\frac{dX_i}{dt} = \left(-\frac{Q_{waste}}{V} * X_i\,\frac{Q_{in}+Q_{recycle}}{Q_{recycle}+Q_{waste}} + \left[\frac{\mu_{max,i}*S}{K_s+S} - b_{e,i}\right] * X_i\right) \qquad \text{SM Eq. 4}$$

where Q_{waste} and $Q_{recycle}$ are the waste and recycle flow rates, respectively, and all other parameters are defined in the main text. Q_{waste} is determined by the MRT:

$$Q_{waste} = \frac{V*Q_{recycle}}{(Q_{in}+Q_{recycle})*MRT-V} \qquad \text{SM Eq. 5}$$

The remaining aspects (e.g., growth parameter combination length, scaling factor, and alpha diversity metrics) are handled identically as in the main text SBR model. Using this approach for MRTs of 1,3,5,7,10, and 15 days, the CSTR model predicted richness values of 22.0, 28.8, 35.5, 41.0, 47.5, and 54.9 and Shannon diversity indices of 3.08, 3.27, 3.39, 3.45, 3.51, and 3.59, both monotonically increasing similar to the SBR model. The full code for the CSTR is include below and can be run by copying into R v3.5.1:

```
library(deSolve);packageVersion("deSolve") #At time of run, 1.21
library(matrixStats);packageVersion("matrixStats") #At time of run, 0.54.0
#==============================================================================================
#Function for Path Length Calculation, taking into consideration ecological limits
#==============================================================================================
    parameter_combination_path_length <- function(MRT, Sstar, Ks, be_constrain, be_eco_min, be_eco_max, 
u_constrain, u_eco_max, u_eco_min){
      be_max_calc  <- c()
      be_min_calc <- c()
      u_max_calc <- c()
      u_min_calc <- c()
      for(i in 1:length(MRT)){

         if(Sstar[i] < 0){
          be_max_calc[i] <- NA
          u_max_calc[i] <- NA
        } else if( ((1+be_eco_max*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i] < u_eco_min){
          be_max_calc[i] <- NA
          u_max_calc[i] <- NA
        } else if( ((1+be_eco_max*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i] > u_eco_max){
          be_max_calc[i] <- (Sstar[i]/Ks*(u_eco_max*MRT[i]-1)-1)/(MRT[i]*(1+Sstar[i]/Ks))
          u_max_calc[i] <- u_eco_max
        } else {
          be_max_calc[i] <- be_eco_max
          u_max_calc[i] <- ((1+be_eco_max*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i]
        }


        if(Sstar[i] < 0){
          be_min_calc[i] <- NA
          u_min_calc[i] <- NA
        } else if( ((1+be_eco_min*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i] > u_eco_max){
          be_min_calc[i] <- NA
          u_min_calc[i] <- NA
        } else if( ((1+be_eco_min*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i] < u_eco_min){
            be_min_calc[i] <- (Sstar[i]/Ks*(u_eco_min*MRT[i]-1)-1)/(MRT[i]*(1+Sstar[i]/Ks))
            u_min_calc[i] <- u_eco_min
        } else {
          be_min_calc[i] <- be_eco_min
          u_min_calc[i] <- ((1+be_eco_min*MRT[i]*(1+Sstar[i]/Ks))/(Sstar[i]/Ks)+1)/MRT[i]
        }
```

```
      }
      result <-
list(be_max_calc=be_max_calc,be_min_calc=be_min_calc,u_max_calc=u_max_calc,u_min_calc=u_min_calc)
    }

#=============================================================================================
#Set the ranges of the ecological umax and be values
#=============================================================================================
      u_eco_max <- 9.8
      u_eco_min <- 0.2
      be_eco_max <- 0.2
      be_eco_min <- 0.02

      Ks   <- 50
   #=============================================================================================
   #Set the parameters of the CSTR
   #=============================================================================================
      Recycle = 192  # Recycle flow rate
      Inflow  = 48  # Inflow flow rate
      Volume  = 12  # Volume of reactor
      S1in    = 250    # Substrate inflow rate
      Ys1S1   = 0.67    # Yield of bacterial growth (fixed) (Henze et al., 1987)
      times   <- seq(0, 100, by =  1) # Reactor operation time (community equilibrium  time)

  #=============================================================================================
  #Calcaulate the waste rate at each MRT based off the parameters and the MRT range
  #=============================================================================================
      #Set the MRT
      MRT <- c(1,1,3,5,7,10,15)

      Waste = Recycle*Volume/(Inflow*MRT+Recycle*MRT-Volume)

 #=============================================================================================
 #Set the constraining umax and be values
 #=============================================================================================
      u_constrain <- 4.5
      be_constrain <- 0.11

      Sstar  <- Ks*(1+be_constrain*MRT)/((u_constrain-be_constrain)*MRT-1)

#=============================================================================================
#Calcaulate the niche path length at each MRT
#=============================================================================================
path <- parameter_combination_path_length(MRT, Sstar, Ks, be_constrain, be_eco_min, be_eco_max,
u_constrain, u_eco_max, u_eco_min)

# Note: Because the values are normalized above, must back transform into the appropriate actual growth
rate here
      Scaling_Factor = 0.25
#============================================================================================
# Calculate the normalized distance of the nice path length at each MCRT
#============================================================================================
      distance <- sqrt( ((path$be_max_calc-
path$be_min_calc)/(path$be_max_calc/2+path$be_min_calc/2)*Scaling_Factor)^2 + ((path$u_max_calc-
path$u_min_calc)/(path$u_max_calc/2+path$u_min_calc/2))^2)

richness <- distance
#============================================================================================
# Select MRTs to use in the differential equations, establish certain required variables
#============================================================================================
    selected_MRT = c(2,3,4,5,6,7)
    RW1      <- c()
    j           <- 1
    diversity <- c()
    bs <- data.frame()
    us <-  data.frame()
    spike_masses <-  c()

#============================================================================================
# For each MCRT selected, run the differential equation and determine the Shannon diversity number
```

```
#==========================================================================================
for (i in selected_MRT){

    print(paste("Evaluating MRT # ", i, sep=""))

    n = round(distance[i]*50, digits = 0)

    print(paste("The number of niches modelled is ", n, sep=""))

    urange = seq(path$u_min_calc[i], path$u_max_calc[i], length.out = n)
    be = (Sstar[i]/Ks*(urange*MRT[i]-1)-1)/(MRT[i]*(1+Sstar[i]/Ks))


#==========================================================================================
#Establish the differential equations for an CSTR AS system as a string to be parsed
#==========================================================================================
    dX__eq  <- paste("dX_", 1:n, " <- (-Qw * X_",  1:n, "* (Qin + Qr)/(Qr + Qw) + u_1_1",  1:n, " *
(So/(Ks + So)) * V * X_",  1:n, " - b_1_1",  1:n, " * V * X_",  1:n, ") / V", sep="")
    dmass_eq <- paste("dmass <- ", paste("dX_", 1:n, sep="",collapse=" + "), sep="")
    dS_eq <- paste("dSo <- (Qin/V * (SS1in - So) - ", paste("u_1_1", 1:n, " * (So/(Ks + So)) * X_", 1:n, "
/ Y_",   sep="",collapse=" - "), ")", sep="")
    state_list <- paste("list(c(dSo, ", paste("dX_",1:n, sep="", collapse = ","), ", dmass))", sep="",
collapse = "")
#==========================================================================================
#Parse the differential equations into a function entitles "Lorenz"
#==========================================================================================
    Lorenz <- function(t, state, parameters){
      with(as.list(c(state, parameters)),{
        eval(parse(text=dX__eq))
        eval(parse(text=dmass_eq))
        eval(parse(text=dS_eq))
        eval(parse(text=state_list))
       })
    }

    parameters500 <- c(b_1_1 = be, Qr=Recycle, Qw=Waste[i], Qin = Inflow, SS1in = S1in, V = Volume,
                       u_1_1 = urange, Ks = Ks, Y_ = Ys1S1)

#==========================================================================================
#Set the initial State
#==========================================================================================
    num_spec <- as.numeric(seq(1,1, length.out = n))
    state <- c(So = 250, X_ = num_spec, mass = n)

#==========================================================================================
#Run the differential equation
#==========================================================================================
    RW1 <- ode(y = state, times = times, func = Lorenz, parms = parameters500)

#==========================================================================================
#Calculate the Shannon diversity metric
#==========================================================================================
    mass_i <- n + 3
    final_X <- n + 2
    nam <- paste("SRT", i, sep="")
    assign(nam, data.frame(be, urange, RW1[99,3:final_X]))
    diversity[j] <- sum(-1*RW1[99,3:final_X]/(RW1[99,mass_i]) * log(RW1[99,3:final_X]/(RW1[99,mass_i])))
    j = j + 1
#==========================================================================================
 }
MRT[2:7]
richness[2:7]*50
diversity
```

Supplemental Table 1. Sequencing Batch Reactor experimental operation conditions and how the model represents thos

Parameter	Experimental Operation	Model Treatment
Innocula	Biomass from the Niederglatt Wastewater Treatment Plant (Niederglatt, Switzerland) aerobic nitrifying reactor	X_0
Substrate	Domestic wastewater	S_{IN} = 250 mg/L
Reactor Full Volume	12 L	V_{full} = 12 L
Operation Mode	Sequencing Batch Reactor (SBR)	SBR
Cycle Time	4 hr	4 hr
Hydraulic Residence Time	12 hr	A total of 4 L of reactor volume is removed per cycle
Sequencing Batch Reactor Cycle	**Fill:** 4 L added over 20 min **React:** 3 hrs of bubbled aeration with mixing **Mixed Drain:** Drain volume of mixed sludge to achieve the MRT*. This occurs over 20 minutes within the react cycle **Settle**: 30 min of settling without aeration **Clarified Drain**: The clarified water is drained to achieve a total of 4 L removed	Q_{fill} = 12 L/hr, t_{fill} = 0.333 hr t_{react} = 3 hr $V_{mixed\ drain}$ = variable, $Q_{mixed\ drain} = V_{mixed\ drain}/t_{mixed\ drain}$, $t_{mixed\ drain}$ = 0.333 hr t_{settle} = 0.5 hr $Q_{clarified\ drain} = (4-V_{mixed\ drain})/t_{clarified\ drain}$, $t_{clarified\ drain}$ = 0.333 hr
Microbial Residence Time (MRT*)	1 d : 2 L $V_{mixed\ drain}$ each cycle 3 d : 2 L $V_{mixed\ drain}$ every other cycle 5 d : 1.2 L $V_{mixed\ drain}$ every third cycle 7 d : 1.7 L $V_{mixed\ drain}$ every sixth cycle 10 d : 1.2 L $V_{mixed\ drain}$ every sixth cycle 15 d : 0.8 L $V_{mixed\ drain}$ every sixth cycle	$V_{mixed\ drain}$ = 2 L each cycle $V_{mixed\ drain}$ = 2 L every other cycle $V_{mixed\ drain}$ = 1.2 L every third cycle $V_{mixed\ drain}$ = 1.7 L every sixth cycle $V_{mixed\ drain}$ = 1.2 L every sixth cycle $V_{mixed\ drain}$ = 0.8 L every sixth cycle
Temperature	Ambient	Not Considered
pH	Experimental Time-point 1: 7.5-8.0 and 2: 7.9-8.3	Not Considered

Supplemental Table 2. 16S rRNA Primer Details

ID	Link Fwd	Primer Sequence Fwd (5'-3')	Link Rev	Primer Sequence Rev (5'-3')	*E. coli* ID	Size (bp)	Regions
B1	CA	AGAGTTTGATCMTGGCTCAG	TG	ATTACCGCGGCTGCTGG	27f-517r	431-506	V1-V3
B2	AT	GTGCCAGCMGCCGCGGTAA	AC	GGACTACHVGGGTWTCTAAT	515f-806R	252-254	V3-V4

Supplemental Table 3. 16S rRNA B1 Primer Library Results

Library ID	Sample Id	Raw	Merged	Primer	Clean	MeanLength	% Surviving	Assigned to ESVs	% Assigned to ESVs
B1cR1-1_Exp1	TP1 1d cDNA 1	148840	128839	126409	124590	475.933	83.71	109357	87.77
B1cR1-2_Exp1	TP1 1d cDNA 2	141997	123321	121074	119356	475.946	84.06	104922	87.91
B1cR1-3_Exp1	TP1 1d cDNA 3	136694	117406	115087	113457	475.737	83.00	100664	88.72
B1cR2-1_Exp1	TP1 3d cDNA 1	118879	103397	101475	100172	471.485	84.26	89776	89.62
B1cR2-2_Exp1	TP1 3d cDNA 2	167028	145179	142519	140757	471.908	84.27	128911	91.58
B1cR2-3_Exp1	TP1 3d cDNA 3	135679	117235	115034	113588	471.43	83.72	103549	91.16
B1cR3-1_Exp1	TP1 5d cDNA 1	109772	93751	92017	90838	472.117	82.75	80906	89.07
B1cR3-2_Exp1	TP1 5d cDNA 2	170236	144327	141610	139970	472.588	82.22	127562	91.14
B1cR3-3_Exp1	TP1 5d cDNA 3	140345	118632	116268	114810	472.277	81.81	105755	92.11
B1cR4-1_Exp1	TP1 7d cDNA 1	154534	130674	128308	126464	477.025	81.84	114199	90.30
B1cR4-2_Exp1	TP1 7d cDNA 2	147855	124450	122201	120539	477.581	81.53	112035	92.95
B1cR4-3_Exp1	TP1 7d cDNA 3	142932	117563	115226	113731	477.534	79.57	105906	93.12
B1cR5-1_Exp1	TP1 10d cDNA 1	102678	85893	84305	83061	479.619	80.89	77349	93.12
B1cR5-2_Exp1	TP1 10d cDNA 2	183737	152123	149373	147212	479.881	80.12	138011	93.75
B1cR5-3_Exp1	TP1 10d cDNA 3	148051	121778	119459	117684	479.833	79.49	110547	93.94
B1cR6-1_Exp1	TP1 15d cDNA 1	105169	85080	83469	82458	479.767	78.41	77182	93.60
B1cR6-2_Exp1	TP1 15d cDNA 2	168924	141298	138717	136667	479.762	80.90	129372	94.66
B1cR6-3_Exp1	TP1 15d cDNA 3	152302	124288	121892	120191	479.72	78.92	113557	94.48
B1gR1-1_Exp1	TP1 1d gDNA 1	177990	152006	149110	143929	468.946	80.86	138142	95.98
B1gR1-2_Exp1	TP1 1d gDNA 2	173932	150198	147317	142366	468.978	81.85	137153	96.34
B1gR1-3_Exp1	TP1 1d gDNA 3	179171	149928	146977	142904	468.671	79.76	136739	95.69
B1gR2-1_Exp1	TP1 3d gDNA 1	187863	168976	165937	164507	451.081	87.57	161137	97.95
B1gR2-2_Exp1	TP1 3d gDNA 2	170712	153803	150907	149561	451.273	87.61	146489	97.95
B1gR2-3_Exp1	TP1 3d gDNA 3	169384	150136	147267	146150	451.248	86.28	142808	97.71
B1gR3-1_Exp1	TP1 5d gDNA 1	143402	110670	108569	108120	450.413	75.40	103698	95.91
B1gR3-2_Exp1	TP1 5d gDNA 2	190782	165792	162767	161876	450.634	84.85	157856	97.52
B1gR3-3_Exp1	TP1 5d gDNA 3	148960	131653	129278	128519	450.931	86.28	125231	97.44
B1gR4-1_Exp1	TP1 7d gDNA 1	212179	185105	181639	180040	455.078	84.85	175021	97.21
B1gR4-2_Exp1	TP1 7d gDNA 2	164239	144878	142124	140921	455.298	85.80	137339	97.46
B1gR4-3_Exp1	TP1 7d gDNA 3	167027	144444	141691	140429	455.397	84.08	136287	97.05
B1gR5-1_Exp1	TP1 10d gDNA 1	144204	62103	60718	60297	459.615	41.81	50646	83.99
B1gR5-2_Exp1	TP1 10d gDNA 2	129958	107649	105550	104406	459.513	80.34	100654	96.41
B1gR5-3_Exp1	TP1 10d gDNA 3	164147	140554	137992	136383	459.588	83.09	131843	96.67
B1gR6-1_Exp1	TP1 15d gDNA 1	176759	154957	152232	150323	459.807	85.04	145376	96.71
B1gR6-2_Exp1	TP1 15d gDNA 2	133794	116622	114548	113345	460.064	84.72	109667	96.76
B1gR6-3_Exp1	TP1 15d gDNA 3	119478	101326	99473	98440	459.871	82.39	94689	96.19
B1pos-1_Exp1	TP1 Positive	122759	96793	94928	92150	486.235	75.07	88269	95.79
B1pos-2_Exp1	TP1 Positive	59922	50863	49802	46664	486.687	77.87	45404	97.30
B1pos-3_Exp1	TP1 Positive	163606	124395	121882	117927	486.128	72.08	112787	95.64
B1cR1-1_Exp2	TP2 1d cDNA 1	91448	81760	80344	80317	476.023	87.83	66833	83.21
B1cR1-2_Exp2	TP2 1d cDNA 2	73627	65807	64684	64656	476.185	87.82	54271	83.94
B1cR1-3_Exp2	TP2 1d cDNA 3	69696	62343	61305	61287	476.138	87.93	51168	83.49
B1cR2-1_Exp2	TP2 3d cDNA 1	47866	42909	42195	42181	476.6	88.12	35854	85.00
B1cR2-2_Exp2	TP2 3d cDNA 2	51369	45966	45219	45204	476.914	88.00	38786	85.80
B1cR2-3_Exp2	TP2 3d cDNA 3	51594	45861	45086	45068	476.859	87.35	38638	85.73
B1cR3-1_Exp2	TP2 5d cDNA 1	61916	54849	53917	53893	476.216	87.04	46062	85.47
B1cR3-2_Exp2	TP2 5d cDNA 2	75099	66602	65461	65435	476.568	87.13	55908	85.44
B1cR3-3_Exp2	TP2 5d cDNA 3	55735	49162	48296	48285	476.383	86.63	40793	84.48
B1cR4-1_Exp2	TP2 7d cDNA 1	79455	70843	69626	69610	477.414	87.61	60671	87.16
B1cR4-2_Exp2	TP2 7d cDNA 2	68544	61081	60051	60042	477.407	87.60	52418	87.30
B1cR4-3_Exp2	TP2 7d cDNA 3	56380	41693	41008	40999	477.3	72.72	34182	83.37
B1cR5-1_Exp2	TP2 10d cDNA 1	84086	75519	74202	74189	477.927	88.23	63169	85.15
B1cR5-2_Exp2	TP2 10d cDNA 2	77703	69307	68117	68104	477.86	87.65	57918	85.04
B1cR5-3_Exp2	TP2 10d cDNA 3	70196	60512	59457	59442	477.802	84.68	50538	85.02
B1cR6-1_Exp2	TP2 15d cDNA 1	81207	71679	70410	70398	479.604	86.69	60701	86.23
B1cR6-2_Exp2	TP2 15d cDNA 2	68524	60301	59255	59245	479.64	86.46	51536	86.99
B1cR6-3_Exp2	TP2 15d cDNA 3	74221	65275	64159	64146	479.606	86.43	55289	86.19
B1gR1-1_Exp2	TP2 1d gDNA 1	80308	71352	70082	70032	473.979	87.20	57137	81.59
B1gR1-2_Exp2	TP2 1d gDNA 2	83048	73418	72056	72005	474.101	86.70	59123	82.11
B1gR1-3_Exp2	TP2 1d gDNA 3	82328	70791	69556	69510	473.922	84.43	57705	83.02
B1gR2-1_Exp2	TP2 3d gDNA 1	59396	53146	52140	51980	471.11	87.51	45329	87.20
B1gR2-2_Exp2	TP2 3d gDNA 2	60586	53809	52812	52631	471.372	86.87	46197	87.78
B1gR2-3_Exp2	TP2 3d gDNA 3	80581	72097	70767	70613	471.131	87.63	62666	88.75
B1gR3-1_Exp2	TP2 5d gDNA 1	49581	43861	43068	42993	470.77	86.71	39200	91.18
B1gR3-2_Exp2	TP2 5d gDNA 2	68528	60695	59491	59389	470.609	86.66	54881	92.41
B1gR3-3_Exp2	TP2 5d gDNA 3	75799	66785	65578	65464	470.693	86.37	60132	91.86
B1gR4-1_Exp2	TP2 7d gDNA 1	65558	58111	57046	57017	476.276	86.97	51122	89.66
B1gR4-2_Exp2	TP2 7d gDNA 2	25288	9991	9767	9760	475.355	38.60	7152	73.28
B1gR4-3_Exp2	TP2 7d gDNA 3	63554	56358	55325	55299	476.32	87.01	49953	90.33

B1gR5-1_Exp2	TP2 10d gDNA 1	86415	77496	76084	76044	475.843	88.00	69804	91.79
B1gR5-2_Exp2	TP2 10d gDNA 2	72367	61713	60601	60537	475.881	83.65	54937	90.75
B1gR5-3_Exp2	TP2 10d gDNA 3	93512	83575	82082	82014	476.003	87.70	75956	92.61
B1gR6-1_Exp2	TP2 15d gDNA 1	76440	67028	65793	65737	476.039	86.00	59719	90.85
B1gR6-2_Exp2	TP2 15d gDNA 2	65236	57340	56315	56278	475.922	86.27	51164	90.91
B1gR6-3_Exp2	TP2 15d gDNA 3	76133	65492	64354	64301	475.737	84.46	58488	90.96
B1-NEG_Exp2	TP2 Negative	68	25	12	12	444.769	17.65	12	100.00
Total		8,379,312	7,132,637	7,000,872	6,928,919		82.69	6,382,207	92.11

Supplemental Table 4. 16S rRNA B2 Primer Library Results

Library ID	Sample Id	Raw	Merged	Primer	Clean	MeanLength	% Surviving	Assigned to ESVs	% Assigned to ESVs
B2cR1-1_Exp1	TP1 1d cDNA 1	111057	107434	105685	105685	253.0	95.16	104323	98.71
B2cR1-2_Exp1	TP1 1d cDNA 2	94108	91064	89602	89602	253.0	95.21	88466	98.73
B2cR1-3_Exp1	TP1 1d cDNA 3	100513	97253	95625	95624	253.0	95.14	94274	98.59
B2cR2-1_Exp1	TP1 3d cDNA 1	96899	93809	92311	92309	253.0	95.26	91146	98.74
B2cR2-2_Exp1	TP1 3d cDNA 2	147762	143384	140941	140939	253.0	95.38	139098	98.69
B2cR2-3_Exp1	TP1 3d cDNA 3	97671	94850	93243	93240	253.0	95.46	92086	98.76
B2cR3-1_Exp1	TP1 5d cDNA 1	91738	88776	87316	87311	253.0	95.17	86356	98.91
B2cR3-2_Exp1	TP1 5d cDNA 2	145781	140827	138661	138657	253.0	95.11	137297	99.02
B2cR3-3_Exp1	TP1 5d cDNA 3	112221	108883	107133	107130	253.0	95.46	106055	99.00
B2cR4-1_Exp1	TP1 7d cDNA 1	102928	99605	98039	98036	253.0	95.25	97132	99.08
B2cR4-2_Exp1	TP1 7d cDNA 2	127090	122334	120398	120393	253.0	94.73	119325	99.11
B2cR4-3_Exp1	TP1 7d cDNA 3	114762	112162	110393	110390	253.0	96.19	109491	99.19
B2cR5-1_Exp1	TP1 10d cDNA 1	117206	112043	110308	110304	253.0	94.11	109527	99.30
B2cR5-2_Exp1	TP1 10d cDNA 2	138460	133306	131396	131387	253.0	94.89	130558	99.37
B2cR5-3_Exp1	TP1 10d cDNA 3	125783	121311	119362	119357	253.0	94.89	118632	99.39
B2cR6-1_Exp1	TP1 15d cDNA 1	138337	132485	130224	130212	253.0	94.13	129225	99.24
B2cR6-2_Exp1	TP1 15d cDNA 2	175836	168618	166069	166055	253.0	94.44	164836	99.27
B2cR6-3_Exp1	TP1 15d cDNA 3	127849	123203	121245	121235	253.0	94.83	120368	99.28
B2gR1-1_Exp1	TP1 1d gDNA 1	131698	117830	110826	110806	253.1	84.14	110281	99.53
B2gR1-2_Exp1	TP1 1d gDNA 2	99830	89948	84411	84396	253.1	84.54	83962	99.49
B2gR1-3_Exp1	TP1 1d gDNA 3	113347	103711	97964	97953	253.1	86.42	97456	99.49
B2gR2-1_Exp1	TP1 3d gDNA 1	139965	129019	121991	121964	253.0	87.14	121597	99.70
B2gR2-2_Exp1	TP1 3d gDNA 2	117613	106626	99848	99828	253.0	84.88	99468	99.64
B2gR2-3_Exp1	TP1 3d gDNA 3	125997	116938	109707	109697	253.0	87.06	109348	99.68
B2gR3-1_Exp1	TP1 5d gDNA 1	141247	131231	124275	124244	253.0	87.96	123827	99.66
B2gR3-2_Exp1	TP1 5d gDNA 2	117368	109572	103850	103825	253.0	88.46	103428	99.62
B2gR3-3_Exp1	TP1 5d gDNA 3	127142	120328	114432	114409	253.0	89.99	114017	99.66
B2gR4-1_Exp1	TP1 7d gDNA 1	144396	133668	124660	124643	253.1	86.32	124181	99.63
B2gR4-2_Exp1	TP1 7d gDNA 2	109825	103071	96516	96503	253.1	87.87	96100	99.58
B2gR4-3_Exp1	TP1 7d gDNA 3	119476	112044	105121	105101	253.1	87.97	104700	99.62
B2gR5-1_Exp1	TP1 10d gDNA 1	159643	147429	138244	138224	253.1	86.58	137772	99.67
B2gR5-2_Exp1	TP1 10d gDNA 2	138183	126399	117728	117701	253.1	85.18	117272	99.64
B2gR5-3_Exp1	TP1 10d gDNA 3	126324	117713	110205	110193	253.1	87.23	109823	99.66
B2gR6-1_Exp1	TP1 15d gDNA 1	161449	148695	139643	139624	253.1	86.48	139021	99.57
B2gR6-2_Exp1	TP1 15d gDNA 2	121930	112455	104549	104537	253.1	85.74	104090	99.57
B2gR6-3_Exp1	TP1 15d gDNA 3	120103	112458	105560	105542	253.1	87.88	105095	99.58
B2pos-1_Exp1	TP1 Positive	125673	117045	114327	114313	252.7	90.96	113946	99.68
B2pos-2_Exp1	TP1 Positive	174729	162894	159177	159158	252.7	91.09	158605	99.65
B2pos-3_Exp1	TP1 Positive	1681597	1566142	1527964	1527771	252.7	90.85	1524061	99.76
B2cR1-1_Exp2	TP2 1d cDNA 1	103801	101585	99747	99747	253.0	96.09	98377	98.63
B2cR1-2_Exp2	TP2 1d cDNA 2	83785	81924	80507	80506	253.0	96.09	79466	98.71
B2cR1-3_Exp2	TP2 1d cDNA 3	72522	71483	70243	70243	253.0	96.86	69465	98.89
B2cR2-1_Exp2	TP2 3d cDNA 1	86889	85385	83806	83805	253.0	96.45	82804	98.81
B2cR2-2_Exp2	TP2 3d cDNA 2	81494	80036	78542	78542	253.0	96.38	77606	98.81
B2cR2-3_Exp2	TP2 3d cDNA 3	64013	63058	61949	61949	253.0	96.78	61262	98.89
B2cR3-1_Exp2	TP2 5d cDNA 1	119412	117342	115329	115326	253.0	96.58	113962	98.82
B2cR3-2_Exp2	TP2 5d cDNA 2	127474	125088	122854	122853	253.0	96.37	121455	98.86
B2cR3-3_Exp2	TP2 5d cDNA 3	120815	119076	116953	116951	253.0	96.80	115564	98.81
B2cR4-1_Exp2	TP2 7d cDNA 1	119252	117218	115164	115163	253.1	96.57	113990	98.98
B2cR4-2_Exp2	TP2 7d cDNA 2	107022	104924	103074	103073	253.1	96.31	101990	98.95
B2cR4-3_Exp2	TP2 7d cDNA 3	83460	82194	80798	80796	253.1	96.81	80054	99.08
B2cR5-1_Exp2	TP2 10d cDNA 1	110965	109021	107033	107033	253.0	96.46	105832	98.88
B2cR5-2_Exp2	TP2 10d cDNA 2	94767	93159	91497	91497	253.1	96.55	90525	98.94
B2cR5-3_Exp2	TP2 10d cDNA 3	107829	106427	104446	104445	253.0	96.86	103393	98.99
B2cR6-1_Exp2	TP2 15d cDNA 1	86686	85152	83692	83692	253.1	96.55	82904	99.06
B2cR6-2_Exp2	TP2 15d cDNA 2	191897	188630	185333	185331	253.1	96.58	183463	98.99
B2cR6-3_Exp2	TP2 15d cDNA 3	106495	104983	103090	103090	253.1	96.80	102136	99.07
B2gR1-1_Exp2	TP2 1d gDNA 1	72516	70523	68169	68169	253.0	94.01	67510	99.03
B2gR1-2_Exp2	TP2 1d gDNA 2	62366	60587	58369	58369	253.0	93.59	57867	99.14
B2gR1-3_Exp2	TP2 1d gDNA 3	75284	72867	70302	70302	253.0	93.38	69594	98.99
B2gR2-1_Exp2	TP2 3d gDNA 1	73891	71416	68528	68528	253.0	92.74	68113	99.39
B2gR2-2_Exp2	TP2 3d gDNA 2	59311	57211	54864	54864	253.0	92.50	54554	99.43
B2gR2-3_Exp2	TP2 3d gDNA 3	72605	69802	66834	66834	253.0	92.05	66429	99.39
B2gR3-1_Exp2	TP2 5d gDNA 1	103791	100694	97264	97263	253.0	93.71	96743	99.47
B2gR3-2_Exp2	TP2 5d gDNA 2	83375	80910	77972	77972	253.0	93.52	77601	99.52
B2gR3-3_Exp2	TP2 5d gDNA 3	109645	106096	102160	102158	253.0	93.17	101566	99.42
B2gR4-1_Exp2	TP2 7d gDNA 1	111611	107887	103392	103383	253.0	92.63	102819	99.45
B2gR4-2_Exp2	TP2 7d gDNA 2	117661	113613	108643	108633	253.0	92.33	108143	99.55
B2gR4-3_Exp2	TP2 7d gDNA 3	127352	122909	117844	117838	253.0	92.53	117186	99.45
B2gR5-1_Exp2	TP2 10d gDNA 1	125615	120989	115704	115703	253.1	92.11	115040	99.43
B2gR5-2_Exp2	TP2 10d gDNA 2	109372	105588	101001	101001	253.1	92.35	100505	99.51
B2gR5-3_Exp2	TP2 10d gDNA 3	107330	103461	99039	99038	253.1	92.27	98450	99.41
B2gR6-1_Exp2	TP2 15d gDNA 1	71655	68957	65926	65925	253.0	92.00	65484	99.33
B2gR6-2_Exp2	TP2 15d gDNA 2	118048	113599	108541	108540	253.0	91.95	107904	99.41
B2gR6-3_Exp2	TP2 15d gDNA 3	131629	126300	120627	120627	253.0	91.64	119804	99.32
B2-NEG_Exp2	TP2 Negative	**174**	91	17	14	236.1	8.05	12	85.71

cNegA_Exp2	TP2 Negative	**839**	619	585	582	252.6	69.37	571	98.11
cNegB_Exp2	TP2 Negative	**114**	28	17	16	238.2	14.04	16	100.00
gM1_Exp2	TP1 Positive	67197	65911	64138	64130	253.0	95.44	63995	99.79
gM2_Exp2	TP1 Positive	48324	47469	46306	46301	253.0	95.81	46184	99.75
gNegA_Exp2	TP2 Negative	**177**	68	43	37	246.5	20.90	31	83.78
gNegB_Exp2	TP2 Negative	**109**	18	7	4	202.4	3.67	4	100.00
Total		10,182,105	9,700,861	9,389,298	9,388,571		92.21	9326618	99.34

Supplemental Table 5. The intermediate processing results of the mRNA annotation pipeline

Sample	Input Reads	Quality Filtered Reads	Pass Quality Reads	% Pass Quality	rRNA Filtered	Non-Filtered (Surviving)	% Surviving Overall	EC Annotated Reads	% Surviving Reads EC Annotated	% Surviving Reads Uniprot-TrEMBL Annotated	Normalization Factor
TP1 1d	51659728	857629	50802099	98.3	7040898	43761201	84.7	9829704	22.5	48.6	2127249485
TP1 3d	41274968	855370	40419598	97.9	7537452	32882146	79.7	5602624	17	38.1	1252836068
TP1 5d	43478158	952651	42525507	97.8	6561606	35963901	82.7	5104415	14.2	31.5	1133057087
TP1 7d	43956036	789388	43166648	98.2	5201824	37964824	86.4	6811803	17.9	38.2	1449679211
TP1 10d	43862862	877647	42985215	98	4458080	38527135	87.8	6253848	16.2	33.8	1302949179
TP1 15d	54367240	1128268	53238972	97.9	5545211	47693761	87.7	7341422	15.4	31.7	1512703018
TP2 1d	49296213	786470	48509743	98.4	11095152	37414591	75.9	7186540	19.2	40.7	1521307202
TP2 3d	51239482	896696	50342786	98.2	13212030	37130756	72.5	6027312	16.2	36.2	1343769486
TP2 5d	41869186	738365	41130821	98.2	6685061	34445760	82.3	5725054	16.6	38.8	1337384189
TP2 7d	51174390	998022	50176368	98	9662159	40514209	79.2	6677636	16.5	36	1457150247
TP2 10d	47099688	978073	46121615	97.9	4918694	41202921	87.5	6990419	17	35.4	1457520368
TP2 15d	47708399	919600	46788799	98.1	8545328	38243471	80.2	6114888	16	33	1260275343

Supplemental Table 6. Results of the Sequencing Batch Reactor multi-species Monod model

MRT	Maximum μ_{max} (day^{-1})	Minimum μ_{max} (day^{-1})	Richness	Number of Niches	Shannon Diversity	TSS (mg/L)	Substrate Uptake Rate (mg-s/[mg-x day])	k_{theo} (day^{-1})
1	5.37	4.63	0.26	13	2.56	141	15.37	5.00
3	5.95	4.05	0.32	16	2.76	397	15.59	5.09
5	6.40	3.60	0.38	19	2.90	561	16.04	5.24
7	6.71	3.29	0.44	22	3.02	668	16.04	5.38
10	7.09	2.91	0.50	25	3.11	795	17.06	5.59
15	7.51	2.49	0.58	29	3.20	902	17.83	5.84